Universidad Zaragoza
1542

Proyecto Fin de Carrera

Reconocimiento visual de juguetes en una mesa de interacción tangible

Autor: Guillermo Navarro Sánchez

Directora: Dra. Sandra Baldassarri

Codirector: Dr. Javier Marco Rubio

Proyecto Fin de Carrera

Ingeniería Informática

Diciembre de 2011

Escuela de Ingeniería y Arquitectura Universidad Zaragoza

Departamento de Informática e Ingeniería de Sistemas Universidad Zaragoza

GIGA Affective Lab

Proyecto Fin de Carrera

Título:	Reconocimiento visual de juguetes en una mesa de interacción tangible
Autor:	Guillermo Navarro Sánchez
DNI:	73004926-M
Titulación:	Ingeniería Informática
Directores:	Sandra Baldassarri Javier Marco Rubio
Departamento:	Informática e Ingeniería de Sistemas
Centro:	Escuela de Ingeniería y Arquitectura
Universidad:	Universidad de Zaragoza
Fecha:	20-Noviembre-2011

Reconocimiento visual de juguetes en una mesa de interacción tangible

Resumen

La idea principal de este proyecto fin de carrera (PFC) consiste en conseguir que los niños muy pequeños puedan interactuar con software y elementos informáticos de una manera más natural a través del uso de juguetes convencionales.

Para lograr este propósito, este PFC se centró en el estudio y ampliación del *framework* de reconocimiento visual para dispositivos *tabletop,* Reactivision. El trabajo se realiza sobre NIKvision, una mesa de interacción tangible especialmente diseñada para su uso por niños pequeños.

En primer lugar se ha realizado un estudio de las limitaciones que presenta Reactivision en el reconocimiento de objetos, para posteriormente, realizar varias fases de análisis e implementación de las distintas funcionalidades necesarias para mejorar las limitaciones analizadas.

La primera fase consiste en mejorar el reconocimiento de fiduciales, marcadores que adheridos a los juguetes permiten su identificación. Esta fase comprende la creación de una nueva colección de fiduciales que permitan ser acoplados a juguetes pequeños y la posterior implementación de los algoritmos para su reconocimiento.

La siguiente fase consiste en la modificación del *framework* Reactivision para permitir la extracción de características geométricas de los objetos planos (juguetes que no lleven fiduciales en su base) reconocidos de manera que aumenten las posibilidades de los juegos a desarrollar.

La tercera fase consiste en la mejora del sistema de comunicación utilizado por Reactivision y los juegos, creando un protocolo de comunicación que permita el envío de volúmenes grandes de datos con estructuras de todo tipo.

La última fase consiste en mejorar la interfaz de usuario de Reactivision, incorporando controles que permitan manejar los parámetros de las nuevas funcionalidades implementadas y en modificar los ficheros de configuración de manera que se puedan automatizar las operaciones de inicialización de parámetros.

A partir de cada una de las mejoras se han creado varios juegos para comprobar su correcto funcionamiento.

Agradecimientos

A Javier y Sandra por sus consejos, conocimiento y paciencia.
A Santiago, Tere, Paula y Elvira por su ayuda en la corrección.

Índice general

Capítulo 1. Introducción

En este primer capítulo, se introduce el proyecto fin de carrera (PFC), explicando primero la motivación inicial, a continuación el contexto en el que se desarrolla, seguido de los objetivos y requisitos que se plantean en este contexto y por último, la estructura que se sigue a lo largo esta memoria.

1.1 Motivación inicial

En la actualidad los sistemas informáticos se han extendido prácticamente a todos los ámbitos. Sin embargo, muchas personas aún muestran reticencia a su uso debido a su manejo difícil y poco natural. Estas dificultades se acentúan en mayor medida en los niños más pequeños (3-6 años), debido a que sus habilidades psicomotrices y cognitivas no están suficientemente desarrolladas como para habituarse al uso de dispositivos de interacción normales en un ordenador, tales como ratones, teclados, etc. Estas limitaciones en sus capacidades hacen que el niño tenga que dedicar mucho tiempo al aprendizaje en el uso de estos dispositivos, lo que provoca una ruptura en la fluidez de las actividades y una disminución de la motivación en utilizar este tipo de tecnología.

El uso de dispositivos con interfaces tangibles puede suponer un cambio en la actitud de las personas frente a las nuevas tecnologías. Estos dispositivos permiten comunicar información al ordenador, igual que otros periféricos como el ratón o el teclado, pero tienen la forma de objetos convencionales y se usan de una forma natural.

Una de las posibilidades que ofrecen las interfaces tangibles, es el uso de juguetes como herramientas para interactuar con el ordenador. Esto incentiva la motivación de los niños más pequeños por usar nuevas tecnologías, ya que lo ven como un juego. Además el uso de juguetes como controles del juego permite a los niños mejorar sus habilidades psicomotrices, a través de la manipulación de objetos [GV01]. Por otra parte el uso de juegos, fomenta que los niños se relacionen entre sí, ya que el juego es común a todos ellos, sin importar su nacionalidad, raza, etc [To02].

Si al uso de interfaces tangibles se añade el uso de *tabletop*s, mesas multitáctiles que pueden ser utilizadas a modo de pizarra o de tablero para los juegos y en las que varios niños pueden jugar utilizando los juguetes o sus manos, se consigue que los niños puedan divertirse de forma conjunta y cooperativa, lo que permite fomentar la relación entre ellos. Esto es muy importante ya que se ha demostrado, que en estas edades el niño forma sus habilidades sociales [Ca02].

En este PFC se ha decidido trabajar con juguetes y mesas de interacción tangible como forma de acercar las nuevas tecnologías a los niños pequeños de manera que su uso les resulte sencillo a través de juegos de ordenador

1.2 Contexto de desarrollo

Este proyecto se basa en el prototipo NIKvision [MCB08] desarrollado por el Grupo de Informática Gráfica Avanzada (GIGA) de la Universidad de Zaragoza. NIKvision es un prototipo de *tabletop* o mesa interactiva, pensada principalmente para que niños pequeños puedan aprender y jugar utilizando un ordenador, a través de juguetes convencionales y no tecnológicos, con los que estén acostumbrados a interactuar. El principal objetivo de este PFC consiste en la ampliación de este prototipo a través del desarrollo de algoritmos que permitan la utilización de nuevos tipos de juguetes y la ampliación de las funcionalidades que NIKvision ofrece de forma que se amplíe del rango de edades para los que están destinados los juegos.

1.2.1 El tabletop NIKvision

El funcionamiento de NIKvision, se basa en el reconocimiento visual con iluminación infrarroja, por medio de una cámara web, de los objetos que se sitúan en la superficie de la mesa. Con este sistema de iluminación, el reconocimiento visual se hace independiente de la iluminación ambiental y al mismo tiempo que no se producen interferencias con la imagen proyectada, al emitirse en distinta frecuencia. Las imágenes capturadas se envían a un software de reconocimiento, que se encarga de tratar las imágenes y extraer la información útil de éstas. Una vez extraída la información, ésta es enviada al juego. Por último la información se representa de forma gráfica a través de un proyector en la superficie de la mesa, que se usa a modo de tablero de juego.

En la figura 1 se puede ver un esquema de la arquitectura hardware de NIKvision:

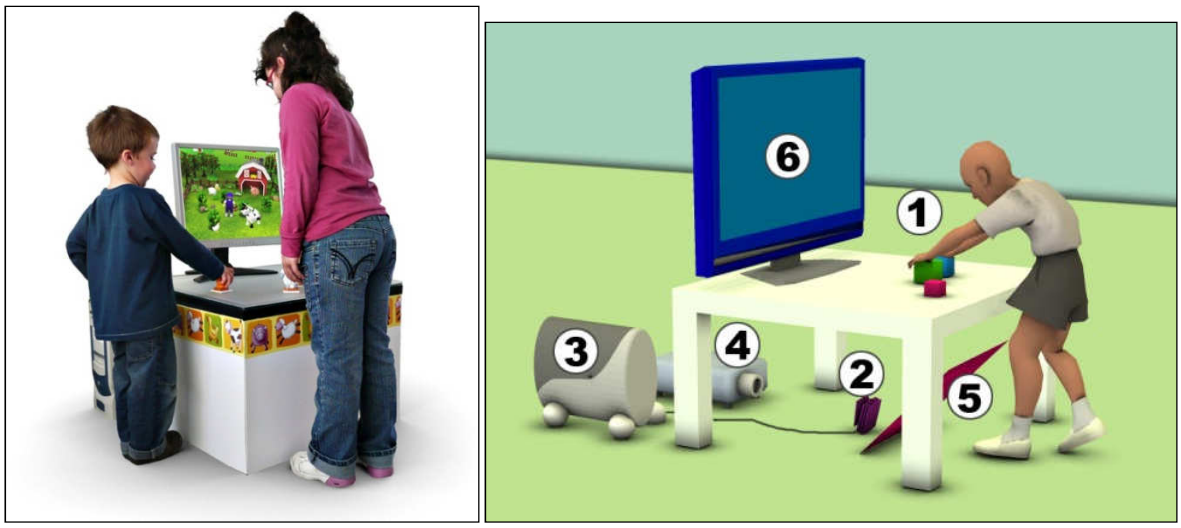


Figura 1 En la izquierda se observa a unos niños jugando con el *tabletop* NIKvision. En la derecha se pueden ver los diferentes elementos que conforman NIKvision.

1. Entradas: Cámara infrarroja con conexión USB (ver fig. 1, img.derecha, elemento 2), encargada de captar los objetos colocados en la superficie de la mesa (ver fig 1, img.derecha, elemento 1) enfocando desde debajo de ésta.

2. Salidas: La imagen activa en la superficie se realiza mediante retroproyección (ver fig. 1, img.derecha, elemento 4), ayudándose de un espejo (ver fig. 1, img.derecha, elemento 5), para que la mesa no necesite excesiva altura (40 cm.) y sea adecuada para la altura de los niños. Adicionalmente, se produce salida gráfica y de audio a través de un monitor de ordenador (ver fig. 1, img.derecha, elemento 6) estándar colocado en un lateral de la mesa, el cual se utiliza para mostrar los entornos gráficos 3D de los juegos.

1.2.2 El framework ReacTIVision

Para la detección de los objetos e interacciones de éstos sobre la superficie de la mesa, NIKvision hace uso del *framework* Reactivision [KB07], una aplicación multiplataforma de código abierto para el prototipado de aplicaciones para dispositivos *tabletop*. Reactivision permite al desarrollador de los juegos abstraerse del acceso al hardware del *tabletop* y de los algoritmos de detección visual de objetos e interacciones del usuario sobre la mesa (ver fig. 2). Esto se consigue comunicando el entorno de desarrollo de la aplicación del *tabletop* (en el caso de NIKvision se usa Adobe Flash), con el *framework* Reactivision, mediante el protocolo de comunicación TUIO, basado en el envío de paquetes UDP [KBBC05]. En el anexo A se describe de forma detallada el funcionamiento de Reactivision y el protocolo TUIO.

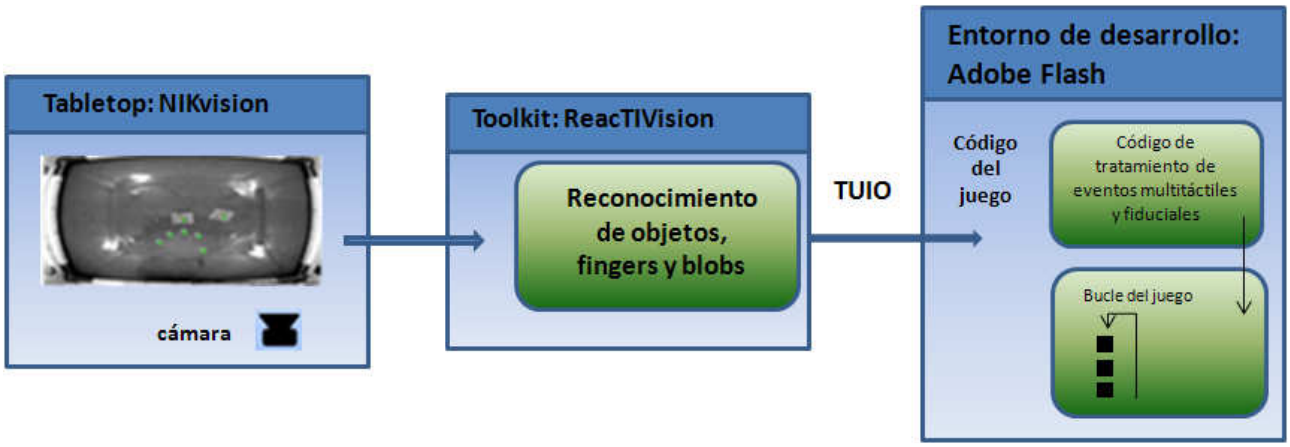


Figura 2 Gráfica de la arquitectura software de NIKvision

Reactivision envía a través de TUIO información de tres tipos de elementos:

- Fiduciales: Reactivision ofrece una librería de marcadores visuales imprimibles (fiduciales), que pueden ser fácilmente pegados a la base de objetos que se vayan a usar en la aplicación del *tabletop*. Cada fiducial tiene una configuración visual diferente, de modo que (análogamente a un código de barras) cada objeto queda unívocamente identificado por la aplicación.
- *Fingers*: Para poder reconocer interacciones táctiles del usuario, Reactivision detecta las puntas de los dedos en contacto con la superficies de la mesa, como áreas blancas circulares de un determinado tamaño y las etiqueta como *fingers* (dedos).
- *Blobs*: Gracias al trabajo realizado en un PFC anterior [En10], Reactivision es capaz de detectar y enviar información de aquellos objetos no identificados por un fiducial, pero

que tampoco se correspondan con puntas de dedos: por ejemplo las palmas de la mano del usuario apoyada sobre la mesa. Reactivision etiqueta estos objetos como *blobs*.

Por tanto, el objetivo general de este proyecto es expandir las capacidades de Reactivision, permitiendo un mejor reconocimiento de interacciones con distintos tipos de juguetes en el *tabletop* NIKvision.

1.3 Objetivos del proyecto

NIKvision está pensado para el desarrollo de juegos para niños en los que puedan utilizar prácticamente cualquier juguete, sin embargo Reactivision posee importantes limitaciones para ser usado con algunos tipos de juguetes, tal y como se describe a continuación:

- Reactivision no detecta correctamente fiduciales pequeños debido a restricciones de resolución del hardware de la cámara usada en NIKVision. Para poder utilizar juguetes pequeños (como por ejemplo, fichas de parchís) no queda otra alternativa que la utilización de juguetes con base circular, de manera que sean reconocidos como *fingers*. El problema es que de esta manera no se puede diferenciar entre los distintos juguetes o fichas. Una posible solución sería utilizar juguetes de diferentes colores, pero aunque a simple vista se puedan identificar, todos ellos seguirían siendo reconocidos como *fingers*, ya que Reactivision no distingue los colores. En la figura 3 se muestra un ejemplo de juguetes de diferentes colores indistinguibles por Reactivision y que son demasiado pequeños para acoplarles un fiducial.

Figura 3 Juguetes de diferentes colores indistinguibles por Reactivision y para los que los fiduciales estándar de Reactivision son demasiado grandes.

La solución utilizada hasta el momento, es pegar los fiduciales impresos en un cartón del tamaño adecuado y posteriormente, este es acoplado en la base de los juguetes de menor tamaño. Esto evita el problema del reconocimiento, sin embargo, limita en cuanto al número de juguetes que se pueden utilizar ya que con un tamaño mayor el número de juguetes que caben en la mesa es reducido. Además empeora la estética del juguete haciéndolo menos atractivo a la vista de los niños, ya que reconocen un elemento extraño en los juguetes, reduciendo así la misión de los interfaces tangibles, es decir, su aspecto deja de ser el de un objeto de uso cotidiano. En la figura 4 se muestra un

ejemplo de juego en el que se puede observar la limitación de los juguetes en cuanto al tamaño de los fiduciales y como éstos son más grandes que la base de los juguetes.

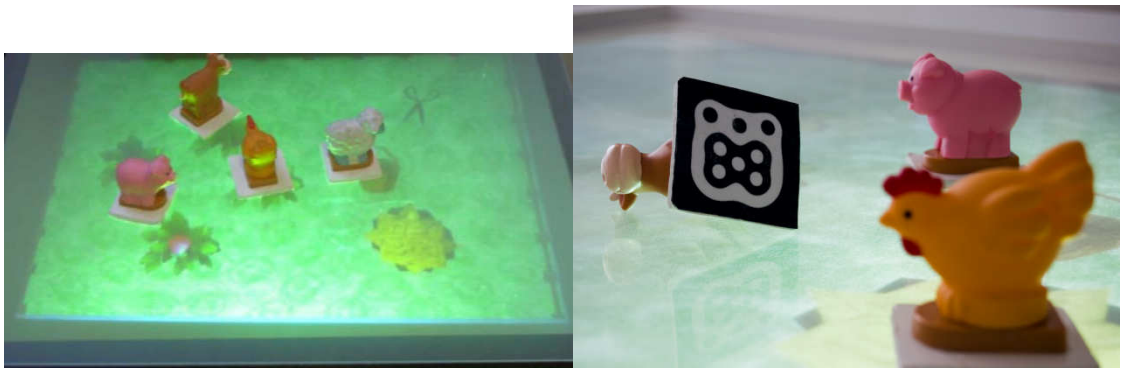


Figura 4 A la derecha se observa el tablero del juego de la granja, con los diferentes juguetes utilizados. A la izquierda se muestra como los fiduciales son notablemente más grandes que la base de los juguetes haciendo que quepan pocos juguetes en la superficie de la mesa.

Se deben por tanto crear nuevos fiduciales que puedan ser acoplados a juguetes pequeños y que puedan distinguirse entre sí.

- Los objetos planos (sin fiducial) son detectados por Reactivision y etiquetados como *Blobs*, sin embargo no se extrae ninguna característica de estos, por lo que las posibilidades de usar juguetes sin fiducial son limitadas, imposibilita por ejemplo el uso de juguetes sin forma definida o deformable en los que es imposible colocar un fiducial, ya que no podrían ser identificados. Este es un problema grave, ya que muchos de los juguetes que los niños pequeños utilizan cumplen esta característica: plastilina, recortables, dibujos de papel, pinceles, etc. Es necesario por tanto la obtención de las características geométricas de los *Blobs* como puede ser el área, la orientación, los agujeros que estos posean, etc de forma que permita identificar juguetes sin fiducial, aumentando así el tipo de juguetes que se puedan utilizar y las posibilidades de los juegos a desarrollar.

- El protocolo de comunicación TUIO limita el tipo y el volumen de datos que se pueden enviar. Al utilizar paquetes UDP de un tamaño fijo, solo permite enviar un número limitado de parámetros, y estos están reservados a la posición del fiducial, las coordenadas de su centro y su orientación. A su vez el tipo de los datos ha de ser fijo no pudiéndose enviar por ejemplo la información de los dibujos hechos por los niños o información con estructuras complejas, como puede ser el contorno de los juguetes. Se debe mejorar por tanto el sistema de comunicación entre Reactivision y los juegos, permitiendo enviar volúmenes grandes de información y ampliando el tipo de datos que es posible enviar.

Por todo ello y dentro del contexto de desarrollo, el objetivo principal de este PFC, es el análisis y estudio de nuevos algoritmos de visión y su posterior diseño e implementación, que permitan ampliar y mejorar el *framework* Reactivision, de manera que se puedan superar sus limitaciones en cuanto al tipo de juguetes reconocidos y la información enviada.

A continuación se definen las necesidades y objetivos específicos de este PFC.

1.4 Requisitos

El primer paso para el desarrollo de este PFC consiste en distinguir el tipo de juguetes cuyo reconocimiento es defectuoso o nulo para determinar los distintos problemas que plantean y de esta forma analizar el tipo de algoritmos de visión necesarios para su correcta detección.

El siguiente paso y una vez analizadas las características de los juguetes a utilizar y sus limitaciones, se procede a realizar el análisis del sistema (Anexo B) para establecer los algoritmos de visión y protocolos de comunicación concretos cuya implementación permita superar los problemas anteriormente citados. Como metodología a emplear se estudiaron dos posibilidades: OMT (Object Modeling Technique) y UML (Unified Modeling Language). Las dos son metodologías de análisis y diseño orientadas a objetos. Al final se optó por la elección de OMT, debido a que sus diagramas se adecuan más a la solución a desarrollar y se determinó que aportaban una visión más detallada del proceso a seguir en la implementación.

A continuación se enumeran los requisitos concretos que se quieren cumplir:

- Reconocer juguetes pequeños: se propone el diseño de una colección de nuevos fiduciales cuya configuración topológica sea más sencilla que la actual, de manera que puedan ser detectados e identificados sin problemas por la cámara de Reactivision, cuando sean impresos en tamaños de entre 2 cm y 4 cm de lado. Esta colección de fiduciales no necesita ser demasiado amplia, ya que se pretende utilizar para identificar fichas de juegos sencillos de tipo parchís o damas en los que el número de jugadores es limitado (en el parchís cuatro y en las damas dos jugadores).

- Extraer otras características (área, contorno, orientación) de aquellos juguetes que no estén identificados mediante fiducial, partiendo de un análisis visual de los *Blobs* asociados a estos objetos. Pudiéndose así usar juguetes deformables como plastilina, recortables, etc. Para ello se estudiaran las características geométricas necesarias para poder identificar este tipo de juguetes y se implementarán los algoritmos para su extracción y cálculo.

- Estudiar y expandir el protocolo TUIO para poder soportar la nueva información obtenida con los nuevos algoritmos de detección que se implementen. Así como crear un nuevo protocolo de comunicación que permita ampliar el tipo de datos a enviar independientemente de su estructura y tamaño.

- Modificar la interfaz de usuario de ReacTIVision para permitir configurar los valores que regulan las nuevas funcionalidades, así como mejorar los ficheros de configuración que permitan cargar y almacenar dichos valores y otros que hasta ahora no se puede.

Para comprobar el correcto funcionamiento de las mejoras llevadas a cabo, se procederá a su incorporación en varios juegos interactivos que las utilicen en el entorno de NIKvision.

1.5 Estructura de la memoria

Esta memoria está dividida en los siguientes capítulos principales:

- Capítulo 1. *Introducción*: Se presentan brevemente la motivación inicial del proyecto, el contexto en el que se enmarca, los objetivos y requisitos que se presentan así como la estructura de la memoria.

- Capítulo 2. *Nueva colección de fiduciales*: Se analiza el diseño de los nuevos fiduciales así como las diferentes alternativas estudiadas para el reconocimiento de juguetes pequeños. A su vez se detallan los algoritmos implementados para su detección y los resultados obtenidos.

- Capítulo 3. *Tratamiento de juguetes sin forma definida*: En este capítulo se explica los problemas y limitaciones que presenta Reactivision en el uso de juguetes deformables y sin forma definida en los que no se puede acoplar fiducial. Se analiza las características que se van a extraer de este tipo de juguetes, porque se han escogido para la identificación de éstos. Por último se detallan los algoritmos implementados para la extracción de estas características y los resultados obtenidos gracias a su cálculo.

- Capítulo 4. *Protocolo de comunicación*: Se explica en qué consiste el protocolo de comunicación diseñado para el envío de la nueva información, así como la estructura de los ficheros XML utilizados.

- Capítulo 5. *Interfaz de usuario*: Este capítulo explica las modificaciones realizadas en la interfaz de usuario de Reactivision así como los ficheros de configuración diseñados para la automatización de las operaciones de inicialización del *framework*.

- Capítulo 6. *Resultados*: En este capítulo se incluyen los resultados de cada implementación, con ejemplos de su funcionamiento y aplicación en diferentes juegos desarrollados para el *tabletop* NIKvision.

- Capítulo 7. *Conclusiones*: Se detallan las conclusiones obtenidas a lo largo del PFC se analiza la valoración personal del proyecto y por último se dan posibles ideas para un trabajo futuro.

La memoria viene acompañada de los siguientes anexos:

- Anexo A. Reactivision y TUIO: Se describe el funcionamiento de Reactivision y TUIO.

- Anexo B. Análisis: Se presenta la metodología de análisis y el desarrollo de las clases que forman el *framework*.

- Anexo C. Alternativas estudiadas: Se analizan en detalle los distintos algoritmos propuestos para superar cada una de las limitaciones anteriormente citadas y se explican las razones por las que varios de ellos fueron rechazados.

- Anexo D. Otros *framework*s y *tabletop*s existentes: En este anexo se comentan otros *frameworks* existentes, realizando una comparación con Reactivision y se analizan algunos *tabletops* del mercado, comparándolos con NIKvision y las funcionalidades que se han desarrollado en este PFC.

- Anexo E. Desarrollo temporal: En este anexo se indica el tiempo empleado en el desarrollo del proyecto en cada una de las fases.

Capítulo 2. Nueva colección de fiduciales

Para que los juguetes puedan ser usados como medio de interacción con juegos de ordenador deben estar marcados con identificadores gráficos. Como se ha explicado anteriormente, los fiduciales que proporciona Reactivision, limitan el tipo de juguetes a utilizar. Por esta razón se ha determinado la necesidad de incorporar nuevos diseños de fiducial para poder adaptarlos a nuevos tipos de juguetes.

En este capítulo se analiza el diseño de los nuevos fiduciales y los algoritmos implementados para su detección. En el primer apartado se detalla el diseño del formato de los fiduciales de la nueva colección. En los siguientes apartados se estudian los pasos de los algoritmos seguidos para la identificación de estos fiduciales:

2.1 “Diseño y detección de nuevos fiduciales”, se detalla el diseño del formato de los fiduciales de la nueva colección.

2.2 “Clasificación de áreas, etiquetado de las áreas que forman el fiducial.

2.3 “Identificación de fiduciales, identificación del tipo de fiducial a partir de las características de las áreas obtenidas.

2.4 “Cálculo de la orientación, cálculo de la orientación del fiducial detectado.

2.5 “Identificación de los fiduciales por pantalla”, identificación gráfica del fiducial.

2.6 “Resultados”, se analizan los resultados obtenidos gracias a la detección de nuevos fiduciales.

En el anexo C se explica en detalle el estudio previo que se realizó y el análisis de alternativas que finalmente llevaron a la creación de la nueva colección de fiduciales.

2.1 Diseño y detección de nuevos fiduciales

El mayor problema que presentaba Reactivision en el reconocimiento de fiduciales pequeños se encontraba en la resolución de la cámara (anexo C). Está era demasiado baja, por lo que detectaba erróneamente las numerosas áreas del fiducial (ver fig. 5). Por tanto se decidió diseñar una nueva colección de fiduciales más sencillos, formados por menos áreas, los cuales gracias a su simpleza no pudiesen dar casos de confusión.

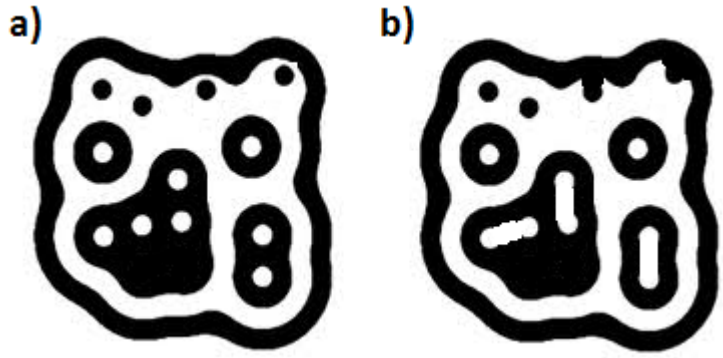


Figura 5 Imagen a): fiducial real. Imagen b: fiducial detectado por la cámara.

La estructura de los nuevos fiduciales, diseñada por ser la más sencilla posible, consiste en un fondo blanco con puntos negros. De esta manera unos fiduciales se distinguirán de otros en función del número de puntos.

Teniendo claro el formato de los nuevos fiduciales, se procedió al estudio de posibles algoritmos de detección de agujeros que pudiesen emplearse para identificar el fiducial. En la figura 6 se puede ver un ejemplo de fiducial detectado, el cual se empleará como ejemplo para mostrar las diferentes fases de los algoritmos implementados.

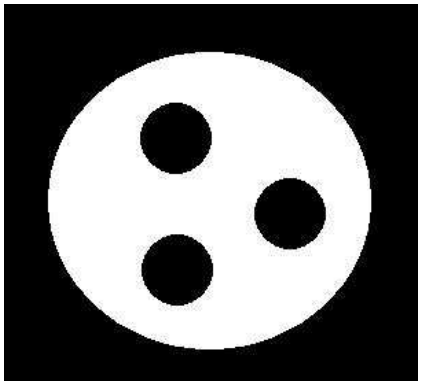


Figura 6 Ejemplo de fiducial de la nueva colección

El reconocimiento de los nuevos fiduciales se basa en el algoritmo de visión de los *fingers* de Reactivision. Este realiza una segmentación de la imagen capturada dividiéndola en regiones blancas y negras y tras descartar las regiones que pertenecen a fiduciales se procede a buscar las áreas que tengan un tamaño similar al de los objetos reconocidos como *fingers*. Este tamaño es establecido a priori por parte del usuario a través de los ficheros de configuración o bien durante la ejecución a través del interfaz de usuario. Una vez detectadas las zonas del tamaño indicado, a través de los algoritmos implementados, se buscan manchas negras sobre el fondo blanco, para comprobar si se trata de un fiducial de la nueva colección o no. Para ello se decide implementar un algoritmo de conectividad entre regiones que permite identificar los agujeros de un objeto y que se detalla a continuación.

2.2 Clasificación de áreas

Una vez detectadas las regiones con el tamaño deseado, se procede a dividirlas en áreas según su color. Para ello en primer lugar se crea una matriz del tamaño de la región detectada, en la que se marca cada celda según el color del pixel que se encuentra en la misma posición en el objeto detectado. De esta manera se tiene una copia del objeto en la matriz y por tanto se puede proceder a distinguir los agujeros del resto de área. En los nuevos fiduciales el fondo está formado por píxeles de color negro igual que los agujeros, mientras que el resto del objeto es de color blanco. En la figura 7 se muestra cómo queda la matriz tras esta primera clasificación.

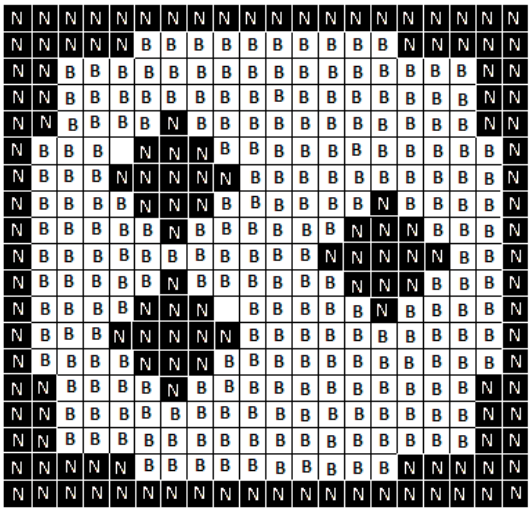


Figura 7 Clasificación de los píxeles (blanco, negro) del objeto detectado.

A continuación es necesario etiquetar todas las áreas que forman el objeto y que son de color negro, ya que entre ellas se encuentran las manchas que interesa identificar. Para ello se recorre la matriz por filas y se busca la primera celda que contenga un pixel negro, este pixel será etiquetado con un identificador, por ejemplo '1'. Tras esto, si algún pixel vecino a este es negro, se etiqueta con la misma etiqueta '1'. Lo siguiente será recorrer toda la matriz para encontrar todos los píxeles marcados con la etiqueta '1' y comprobar si sus vecinos también son de color negro, en tal caso serán marcados de la misma forma que su vecino. Una vez finalizado el etiquetado de todos los píxeles vecinos, se vuelve a recorrer la matriz en busca del siguiente pixel negro sin etiquetar y se vuelve a repetir los mismos pasos que se han explicado pero cambiando el nombre de la etiqueta, por ejemplo ‘2’. El algoritmo finalizará cuando todos los píxeles negros hayan sido etiquetados. En la figura 8 se observa un ejemplo de cómo quedaría la matriz tras ser etiquetada y en la que se han detectado 4 áreas distintas.

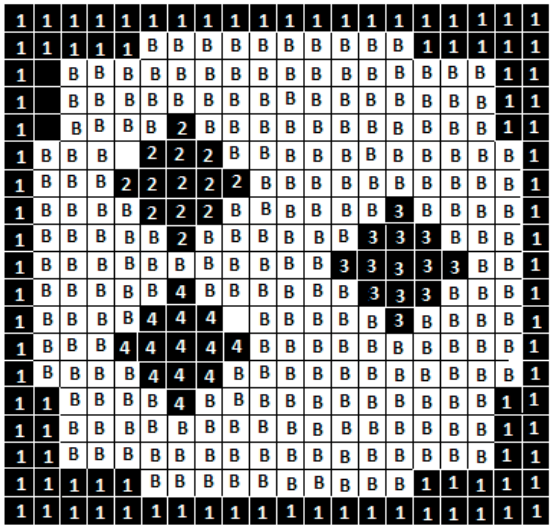


Figura 8 Clasificación de las áreas negras del objeto detectado.

2.3 Identificación de fiduciales

Una vez etiquetadas las zonas negras que contiene el objeto detectado, el siguiente paso consiste en discriminar las regiones del exterior de la zona blanca, ya que estas no forman parte del fiducial. Para ello se recorre la matriz por filas comprobando todas las áreas etiquetadas. Si estas contienen algún pixel cuya coordenada 'x' o 'y' es igual a la anchura o altura del objeto respectivamente son descartadas, ya que esto indicará que se encuentran por fuera de la zona blanca que forma el fiducial y por tanto no pertenece a este.

A continuación se procede al estudio de las áreas negras restantes. Estas pueden ser tanto los agujeros que forman parte del fiducial como áreas causadas por ruido de cámara. Por tanto se deben descartar las producidas por ruido quedándose sólo con los agujeros. Para ello el algoritmo recorre de nuevo la matriz y para cada una de las áreas etiquetadas cuenta el número de píxeles que las forman. Una vez contado el número de píxeles que forman cada área, se escogen las tres de mayor tamaño que pasan a ser las candidatas a ser posibles agujeros del fiducial. Las áreas más pequeñas y por tanto descartadas, corresponden a ruido de cámara, el cual solo está formado por unos pocos píxeles entre 3 y 10, muchos menos píxeles que los agujeros del fiducial. En la figura 9 se muestra un ejemplo ilustrativo de la diferencia entre las áreas que forman los agujeros del fiducial y las áreas que corresponden a ruido de cámara.

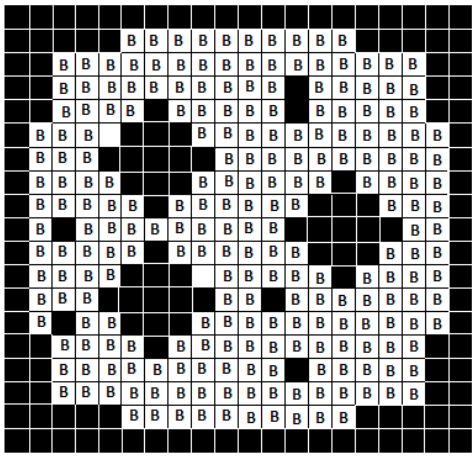


Figura 9 Ejemplo de fiducial detectado con ruido. Los agujeros del fiducial, son notablemente más grandes que el ruido de cámara.

No obstante puede ser que alguna de las 3 áreas escogidas, pese a ser de las de mayor tamaño, sea producida por ruido. De darse el caso de que estas 3 áreas correspondiesen solamente con ruido, el algoritmo estaría reconociendo erróneamente ruido de cámara como fiduciales de la nueva colección ya que estaría confundiendo el ruido con agujeros del fiducial. En la figura 10 se puede ver un ejemplo de un posible error de identificación. Para asegurar que se trata de agujeros y no de ruido se comprueba que el número de píxeles que forman estas tres áreas cumplen una serie de restricciones de tamaño. Para que el objeto reconocido sea identificado como fiducial, debe cumplir que el tamaño de sus agujeros esté comprendido entre un 3% y un 60% del área total del objeto. En caso contrario se considera que los agujeros son debidos a ruido y no serán descartados del proceso de búsqueda de agujeros. A continuación se detallan el algoritmo seguido para la clasificación de agujeros:

Si: el tamaño de cada uno de los tres agujeros ∈ [3%,60%] del tamaño del área del fiducial
entonces
Es un fiducial con tres puntos negros

sino si: el tamaño de dos de los tres agujeros [3%,60%] del tamaño del área del fiducial
entonces
Es un fiducial con dos puntos negros

sino si: el tamaño de uno de los tres agujeros [3%,60%] del tamaño del área del fiducial
entonces
Es un fiducial con un punto negro

sino
El objeto detectado no corresponde con un fiducial de la nueva colección

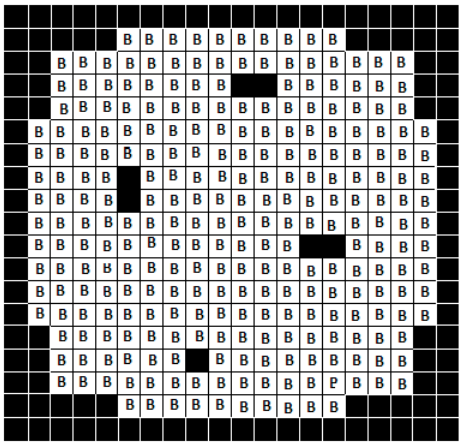


Figura 10 Objeto con ruido, en el que las 3 áreas mayores corresponden a ruido.

Tras verificar el funcionamiento del algoritmo, se ha identificado que las comprobaciones previamente realizadas (tamaño del fiducial y tamaño de los agujeros) no eran suficientes. Se producían demasiados falsos positivos debido al ruido de cámara, lo que hacía poco preciso el sistema de reconocimiento de los nuevos fiduciales. Al observar el tipo de ruido que producía falsos positivos, se detectó que lo que lo diferenciaba respecto de los fiduciales era el número de áreas que se reconocerían en la clasificación. El ruido que provocaba los errores de reconocimiento estaba formado por un gran número de áreas, es decir, un gran número de agujeros o puntos negros, mientras que los fiduciales reales sólo están formados por unas pocas áreas (los agujeros y algún pixel producido por ruido). En la figura 11 se muestra un ejemplo del tipo de ruido que provocaba falsos positivos.

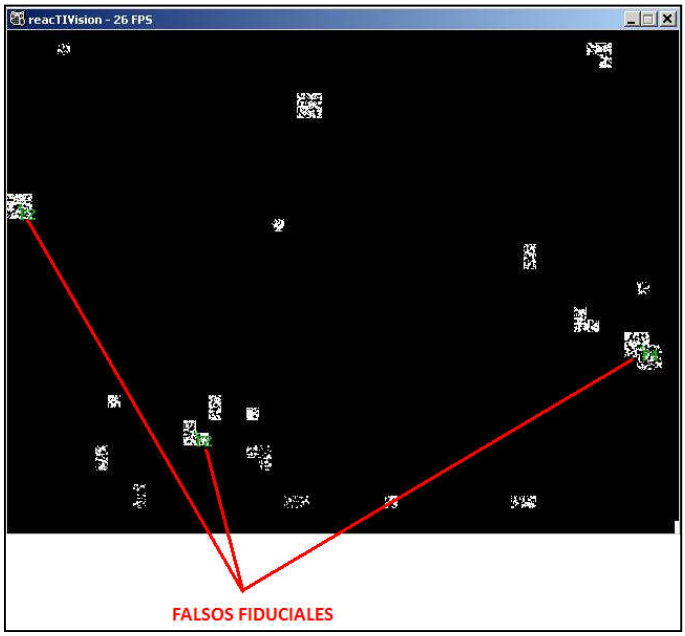


Figura 11 Imagen de una superficie sin objetos en la que solo se ha capturado ruido de cámara. El ruido provoca falsos positivos de fiduciales al estar formado también por una zona blanca con agujeros negros.

Por tanto se estableció una restricción que consistía en comprobar que el número de áreas detectadas fuese pequeño para que el objeto reconocido fuese considerado como fiducial y no como ruido. El resultado fue una reducción considerable del número de falsos positivos, pero aun así no era suficiente, se seguían produciendo demasiadas falsas identificaciones, lo que hacía poco fiable el sistema de reconocimiento.

Tras estudiar de nuevo el ruido de cámara que estaba produciendo los falsos positivos, se identificó que los fiduciales se diferencian de este en la zona blanca que los formaba, es decir, los fiduciales están compuestos por muchos más píxeles blancos que el ruido de cámara detectado. Por tanto se estableció como restricción, que el número de píxeles blancos tuviese que superar un determinado umbral, definido por el usuario, para que el elemento reconocido se considere como un fiducial. Tras volver a probar el sistema de reconocimiento, se dio por válido, pues había reducido casi por completo el número de falsos positivos.

2.4 Cálculo de la orientación

Para determinados juegos es interesante disponer de la orientación de los nuevos tipos de fiducial. Se estudió entonces como dotar a los fiduciales de esta característica y se optó por establecer los agujeros en el fiducial de forma que no estuviesen centrados. De esta manera, es posible calcular un vector con origen en los píxeles blancos y final en los píxeles negros, del cual se puede obtener fácilmente la orientación.

Para calcular la orientación se procedió a determinar el centro de gravedad de la zona blanca y el centro de gravedad de la zona negra para después hallar la orientación del vector que va desde el centro blanco al centro negro. Para obtener el grado de orientación de vértice se ha utilizado la siguiente fórmula:

$$orientación = \frac{1}{\tan((cY\_n - cY\_b) \ / \ (cX\_n - cX\_b))}$$

Siendo (cX_n, cY_n) y (cX_b, cY_b) las coordenadas de los centros negro y blanco respectivamente. En la figura 12 se observa un fiducial de la nueva detección detectado y los valores de su orientación.

Figura 12 Fiducial reconocido y ventana de Reactivision mostrando los valores del ángulo en el que está orientado.

2.5 Identificación de fiduciales por pantalla

Por último se procedió a realizar la representación gráfica de los fiduciales para poder determinar claramente el tipo de fiducial del que se trata y poder identificarlo de forma rápida. Para ello se modificó el sistema de visualización que Reactivision implementa para la visualización de elementos detectados. El método utilizado por Reactivision consiste en la superposición de letras identificativas en cada uno de los objetos identificados, de manera que pueda reconocer de forma inequívoca el tipo de objeto del que se trata.

El sistema de visualización de Reactivision muestra los objetos detectados con un número identificativo en cada uno de los fiduciales y una F en los objetos identificados como *fingers*. De esta manera no era posible distinguir entre los nuevos tipos de fiduciales creados, por lo que se optó por establecer una nueva terminología para identificar los nuevos fiduciales, utilizando el símbolo: 'F2' para los fiduciales de un agujero, 'F3' para los de dos agujeros y 'F4' para los de tres agujeros. Se modificó entonces la función `drawObject`, de manera que cuando se reconoce un nuevo fiducial, se muestra la etiqueta F2, F3 o F4 sobre éste para identificarlo según el tipo. En la figura 13 se muestra la representación de los tres nuevos tipos de fiducial.

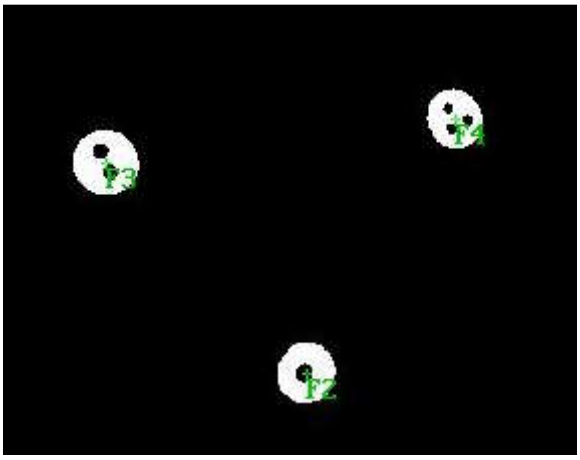


Figura 13 Tres nuevos tipos de fiduciales identificados según el número de agujeros.

2.6 Resultados obtenidos

Como resultado del diseño de los nuevos fiduciales y de la implementación del nuevo algoritmo de reconocimiento han sido superadas varias de las limitaciones que presentaba el *framework* Reactivision.

En un primer lugar el diseño de los nuevos fiduciales permite crear identificadores gráficos de diferentes formas, no tiene que ajustarse a ninguna forma en concreto, tan solo deben cumplir las restricciones de tamaño y proporciones de área blanca que se han establecido. De este modo se supera la limitación en cuanto a la forma de los fiduciales, ya que se comprobó que cuanto más pequeño era el fiducial más cuadrado debía ser para que la detección fuese correcta. En la figura 14 se muestra un ejemplo de fiduciales de la nueva colección impresos en diferentes formatos y juguetes en los que se pueden acoplar adaptándose perfectamente a su base. Notar que simplemente se debe cumplir que el gráfico adherido al juguete sea blanco y que posea en su interior manchas negras que cumplan con las restricciones de tamaño establecidas.

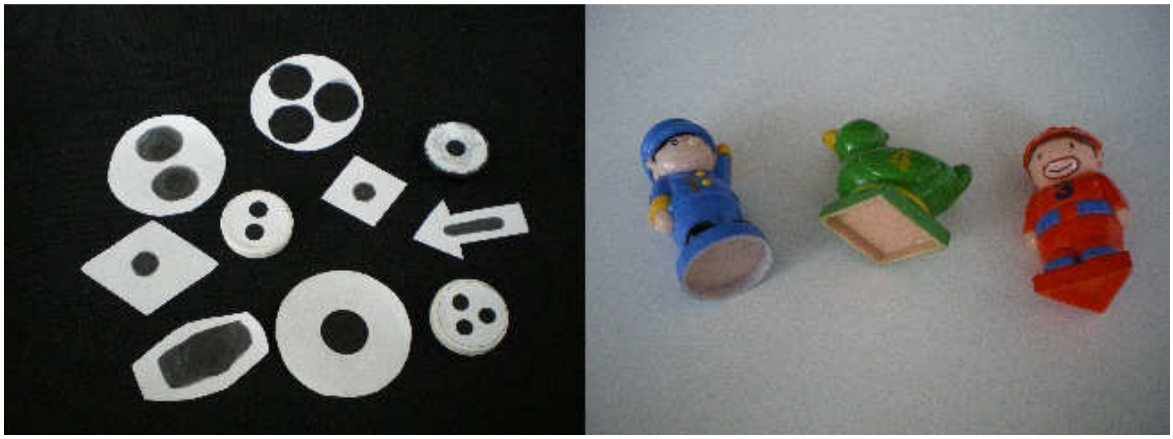


Figura 14 Izquierda: nuevos fiduciales con diferentes formatos. Derecha: juguetes en los que se puede acoplarlos.

En la figura 15 se puede observar un ejemplo de cómo son reconocidos en Reactivision un ejemplar de cada tipo de fiducial de la nueva colección y como se adaptan perfectamente a fichas como las del parchís haciéndolas distinguibles. Sin los nuevos fiduciales todas ellas eran reconocidas como *fingers* por lo que no podían utilizarse para distinguir a un jugador de otro.

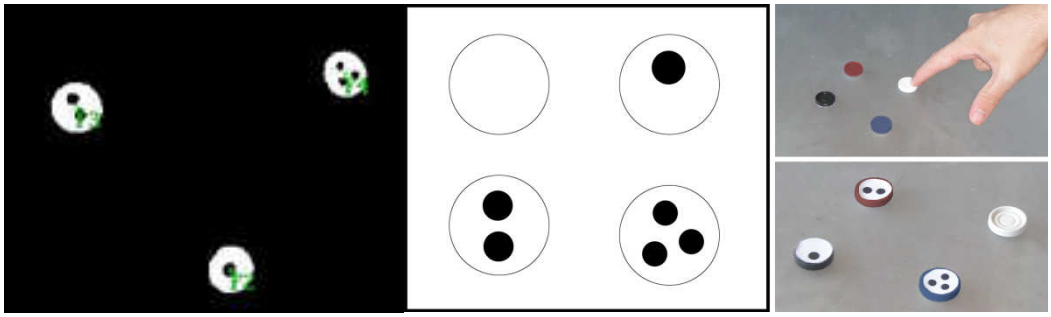


Figura 15 Izquierda: nuevos fiduciales reconocidos por Reactivision. Derecha: fiduciales adaptados a fichas el parchís.

Por otro lado gracias a la sencillez de los fiduciales y al algoritmo de reconocimiento, el usuario puede establecer el tamaño de los fiduciales, mediante la tecla 'f' y las flechas de dirección, no teniendo porque ser de un tamaño fijo y pudiendo reducir en gran medida el tamaño mínimo que Reactivision reconoce. Esto permite utilizar juguetes mucho más pequeños y, por tanto, se pueden utilizar muchos más juguetes a la vez ya que por el gran tamaño de los fiduciales que Reactivision reconocía, el número de juguetes que se podían utilizar a la vez era reducido ya que ocupaban enseguida toda la superficie del *tabletop*. Además gracias a esta reducción de tamaño el resultado de los juguetes es estéticamente más agradable. Como se puede observar en la figura 16 el tamaño mínimo de los nuevos fiduciales se ha reducido en torno a un 75% respecto al de los reconocidos hasta el momento.

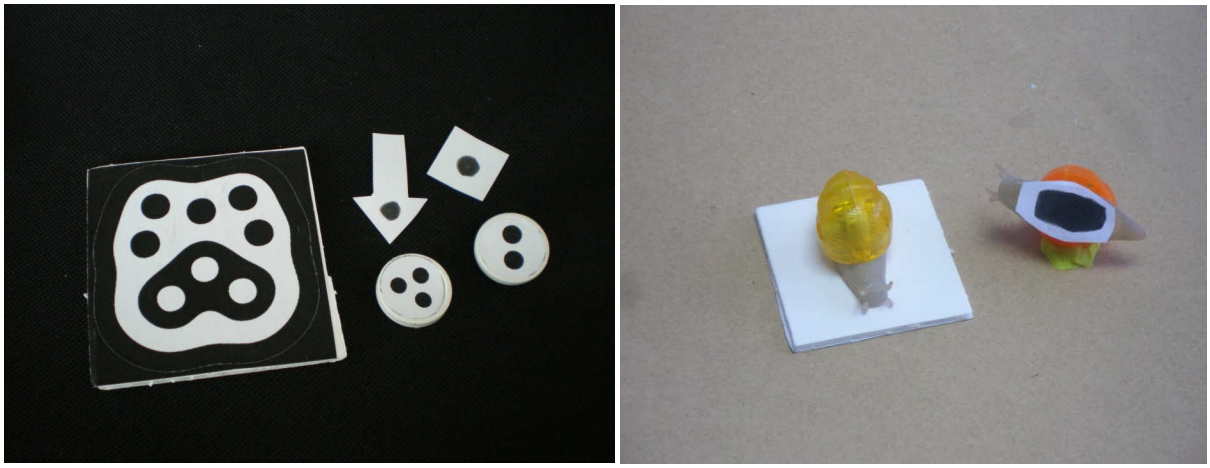


Figura 16 Izquierda: Comparación de un fiducial estándar de Reactivision con cuatro fiduciales de la nueva colección. Derecha: Comparación de un juguete aumentado con un fiducial estándar de Reactivision y el mismo tipo de juguete aumentado con un fiducial de la nueva colección.

Como valor añadido se ha conseguido una reducción en el tamaño de los fiduciales simplemente mejorando los algoritmos de reconocimiento y creando una nueva colección de fiduciales, de forma que no se ha requerido de la inversión en nuevas cámaras o sistemas más precisos.

Capítulo 3. Tratamiento de juguetes sin forma definida.

Los niños suelen utilizar en sus juegos una gran cantidad de objetos, muchos los cuales son juguetes a los que es posible adaptarles un fiducial y así poder utilizarlos en un *tabletop*, sin embargo, esto no siempre es posible. Los niños por su gran imaginación convierten cualquier objeto en un juguete y no a todos estos objetos es posible adaptarles un fiducial, ya sea debido a los materiales con los que están construidos o porque no tiene una forma definida. Ejemplos de estos podrían ser plastilina, recortables hechos por ellos mismos, bolas de papel, etc. Para superar esta limitación en un PFC anterior [En10] se implementó el reconocimiento de objetos sin un fiducial asociado. Sin embargo, de estos objetos no se extrae ninguna característica, solo se detecta y se informa de su posición, lo que limita en gran medida las posibilidades que éstos ofrecen. Por ello es necesario extraer características geométricas de este tipo de objetos de forma que puedan desarrollarse juegos que aprovechen todo su potencial.

En este capítulo se explica el proceso de distinción de objetos sin fiducial de los demás objetos de Reactivision para después explicar cuáles son las características que se han decidido extraer de los objetos sin fiducial, así como el proceso de extracción y posterior cálculo de las mismas. Además se muestran los resultados obtenidos con estos nuevos diseños. En el anexo C se pueden ver los distintos algoritmos estudiados y que finalmente han sido descartados.

3.1 Diferenciación de objetos

El *framework* de reconocimiento Reactivision, como se ha explicado previamente, detectaba objetos marcados con fiduciales, objetos circulares del tamaño de una yema del dedo y objetos sin fiducial. Sin embargo a la hora de la representación gráfica de los dedos y objetos sin fiducial, no se producía distinción entre ellos. Ambos eran identificados con la letra 'F' (de la palabra *finger*, dedo en inglés). Esto provocaba confusión entre los usuarios que no distinguían entre unos objetos y otros al visualizarlos en pantalla. En la figura 17 se observa un ejemplo de este error.

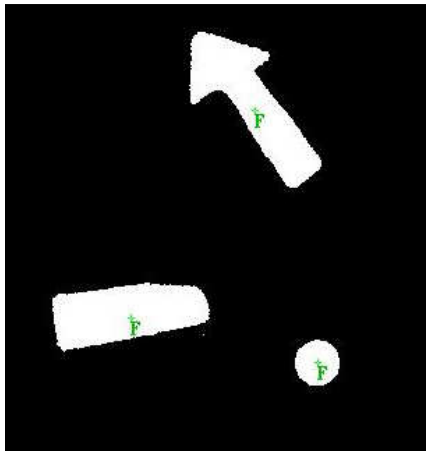


Figura 17 Objetos sin fiducial (flecha y rectángulo) identificados con la misma letra que los dedos de las manos (círculo) 'F'.

Para solucionar este problema se modificó la función que representaba gráficamente la letra 'F', encima de cada elemento reconocido, de manera que sobre los objetos sin fiducial se muestra la letra 'B' (de la palabra *blob*, mancha en inglés). De esta manera cada tipo de elemento es fácilmente identificable a través de estas letras. En la figura 18 se muestra como ya se hace una distinción entre elementos y por tanto se pueden distinguir a simple vista.

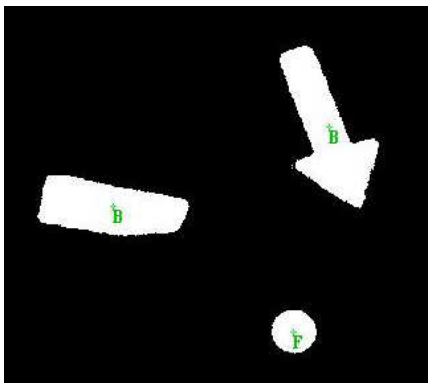


Figura 18 Objetos sin fiducial correctamente identificados con la letra 'B' y *fingers* con las letra 'F'

Otro problema más grave a la hora del reconocimiento, era que se producían falsos positivos en el reconocimiento de objetos sin fiducial. Éstos eran confundidos con partes del dibujo que forma el fiducial. Por ello, en el momento en que se necesitaba trabajar con objetos de varios tipos Reactivision los confundía. En la figura 19 se puede observar un caso de falsos positivos a la hora de reconocer un fiducial. Este problema ocurre ya que una vez reconocidos todos los tipos de elementos: *fiduciales*, *fingers* y *blobs*, no se hacía ninguna comparación de la posición entre los diferentes objetos. Es decir, si por ejemplo se había reconocido un fiducial en las coordenadas (x,y), a la hora de reconocer posibles blobs, no se comprobaba si estos eran detectados en las mismas coordenadas (x,y) del fiducial, por lo que se podía estar detectando el mismo elemento como fiducial y como blob. Lo mismo ocurría en el caso de los fingers.

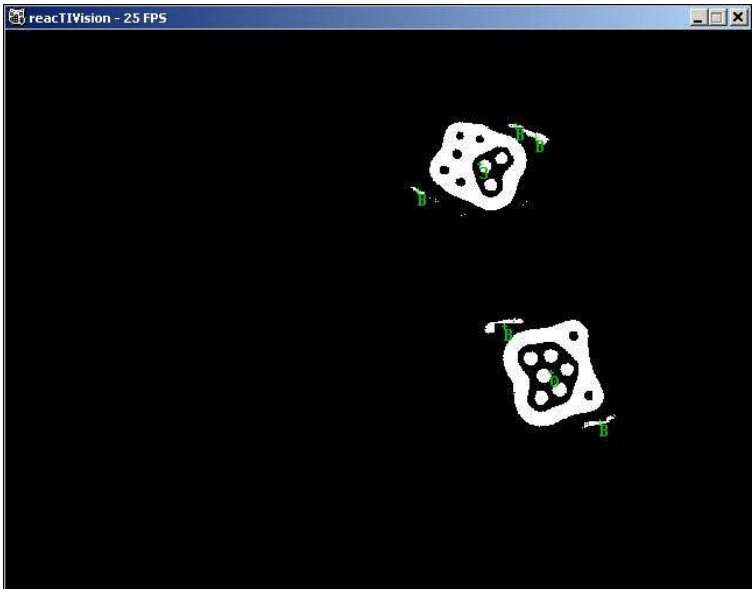


Figura 19 Falsos positivos de *blobls.* Los elementos pequeños de color blanco que forman los fiduciales son confundidos con *blobls.*

Para solucionar este problema se modificó el algoritmo de reconocimiento de manera que comprobase la posición de cada elemento detectado. Primero, se detectan los fiduciales, y al reconocer uno nuevo se comprueba que la distancia desde su centro al centro de cualquier otro fiducial reconocido sea mayor que la longitud de uno de sus lados. De esta manera se asegura que el nuevo objeto reconocido, no se encuentra sobre un fiducial ya reconocido y por tanto no se trata de un falso positivo.

Después se procede a la identificación de posibles *fingers*. Cada vez que se detecta un finger se realiza la misma comprobación anterior y además se comprueba que la distancia entre centros con los fingers ya reconocidos sea mayor que su tamaño.

Por último, se reconocen los objetos sin fiducial y se realiza una comprobación análoga a las explicadas, comparando los centros de estos objetos con el de los fiduciales y dedos identificados. En la figura 20 se muestra como se han eliminado los falsos positivos.

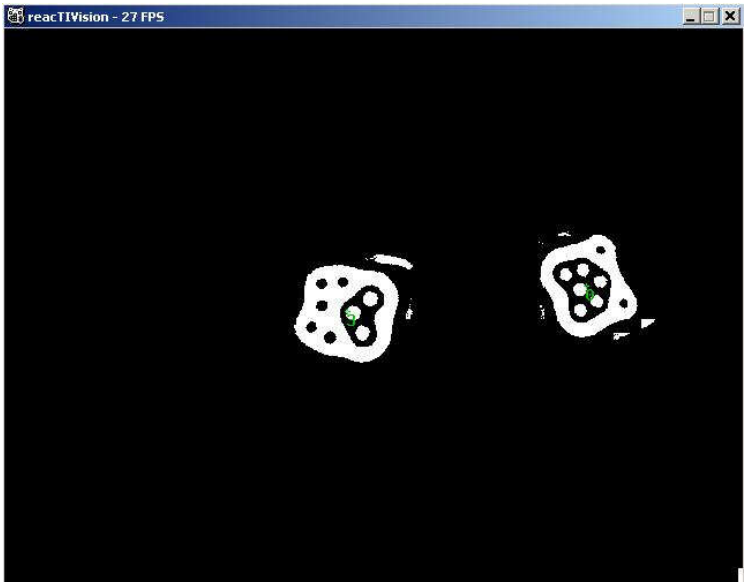


Figura 20 Fiduciales correctamente identificados, sin falsos positivos.

3.2 Elección de características

Como primera tarea se procedió al estudio y elección de las características que era más interesante extraer de los objetos sin fiducial, siempre teniendo en cuenta que su uso estaba destinado a juegos con niños muy pequeños. Había que tener en cuenta que las características extraídas se debían poder calcular con la información proporcionada por imágenes, ya que los juguetes utilizados, como se comentó en el capítulo 1, son juguetes convencionales sin ningún dispositivo tecnológico.

Las diferentes características que se consideró que podían ser útiles en el diseño y desarrollo de juegos interactivos fueron: área, orientación, velocidad de movimiento, velocidad

de rotación y contorno del objeto. A continuación se nombran las diferentes características estudiadas, cuáles fueron finalmente escogidas y la razón de su elección.

La **orientación** ya era extraída en los juguetes con fiducial y para muchos de los juegos que se habían desarrollado era imprescindible el uso de ella. Sirve tanto para saber hacia dónde está mirando el niño con el juguete o para utilizar algún objeto como una ruleta, subir y bajar volumen, etc. Por ello esta fue una de las características escogidas.

El **área** fue otra de las características elegidas, ya que en determinados juegos puede ser importante saber el tamaño del objeto. Por otra parte esta característica permitiría conocer la presión que el niño está haciendo en todo momento algo que podría ser interesante en juegos que requiriesen una información más precisa, como por ejemplo en un juego en el que los niños pintasen con un pincel y dependiendo de la presión el trazo fuese más fino o más grueso.

Las características de **velocidad,** tanto de movimiento como de rotación, finalmente fueron descartadas. Estas características requerirían un manejo de los juguetes con una mayor precisión y los niños pequeños tienen sus habilidades psicomotrices poco desarrolladas, por lo que tras analizarlo se llegó a la conclusión que estas características serían irrelevantes o poco útiles en el desarrollo de juegos para niños tan pequeños.

Por último el cálculo de los **contornos** de los juguetes es interesante y útil. Los juguetes utilizados pueden tener formas muy diversas y no se tiene más información de ellos que su posición. Por ello era preciso disponer de esta característica para poder aprovechar al máximo el juguete utilizado. Por ejemplo se podría detectar el contorno de los juguetes empleados para crear un juego en el que los niños tuviesen que construir murallas de un castillo o colorear el interior de las siluetas de sus juguetes.

A continuación se explica en detalle el cálculo de las tres características seleccionadas para su implementación: área, orientación y contorno.

3.3 Cálculo del área de un juguete

Para el cálculo del área y la orientación se usa la información que Reactivision proporciona de los objetos sin fiducial. El mecanismo de detección de estos objetos se basa en la misma técnica de detección que los *fingers*, tras haber segmentado la imagen se procede a la búsqueda de todas las regiones de color blanco que tengan un tamaño similar al establecido en los ficheros de configuración. Una vez detectadas, para cada una de ellas, Reactivision devuelve una matriz en la que se almacena la imagen de cada región, las coordenadas de su posición y la altura y anchura de ésta. Con la altura y la anchura se puede conocer el área rectangular dentro de la que se encuentra el objeto, sin embargo por su estructura irregular, este tipo de juguetes normalmente no ocupará toda esta área. Por tanto aprovechando la matriz devuelta con la imagen de la región y dado que el juguete corresponderá al área de la imagen formada por píxeles blancos, se cuenta el número de píxeles blancos encerrados en el área en la que se encuentra el objeto. De esta manera se obtiene el área de un juguete sin fiducial de una forma precisa. La fórmula utilizada es la siguiente

$$area = \sum_{i=0}^{anchura-1} \sum_{j=0}^{altura-1} g(i,j)$$

donde *g(i,j)* representa el color del pixel situado en las coordenadas i,j.

3.4 Cálculo de la orientación de un juguete

Los juguetes de los cuales se necesita obtener la orientación (plastilina, recortables, muñecos) son muy diferentes entre sí y no tienen ninguna característica en común, lo único que los hace similares son su tamaño aproximado y que su imagen estará formada en su totalidad por píxeles blancos. Para el cálculo de la orientación de los fiduciales se aprovecha la distribución de las áreas blancas y negras, por ello es imposible utilizar el mismo método en los objetos sin fiducial ya que están formados en su totalidad por píxeles blancos. Por lo tanto para su cálculo, es necesario buscar algoritmos que sólo dependiesen de los descriptores de forma del objeto como pueden ser el perímetro, elongación, momentos de la imagen, etc. Para la elección de este algoritmo se han estudiado y analizado numerosas fuentes: [GW08], [MA02], [SD04], [SBBKT02], [Go06], [De01].

Todas ellas determinaban que la orientación se suele calcular haciendo uso de los momentos geométricos del objeto. Los momentos son propiedades numéricas que se pueden obtener de una determinada imagen, tienen en cuenta todos los píxeles de la imagen y no sólo los bordes. Los momentos de una imagen se clasifican en simples, centrales y centrales normalizados. En este caso se han utilizado los momentos centrales normalizados, los cuales permiten reconocer objetos dentro de una imagen independientemente de cuál sea su posición o tamaño. En concreto para el cálculo de la orientación se utilizan los momentos de segundo orden los cuales reflejan la distribución de masa de un cuerpo, respecto a un eje de giro. Los momentos de segundo orden responden a la ecuación:

$$U(p,q) = \sum_x \sum_y (x-\bar{x})^p (y-\bar{y})^q g(x,y)$$

donde $\bar{x}$, $\bar{y}$ el centro de gravedad del *blob* y *g(x,y)* representa el color del pixel situado en las coordenadas *i,j* y donde *p* y *q* son los valores que identifican el tipo de momento, para los momentos de segundo orden se debe cumplir p+q=2. Los tres momentos centrales de segundo orden forman los componentes del tensor de inercia o matriz de rotación, siendo *U(0,2), U(2,0)* y *U(1,1)* los tres momentos de segundo orden.

$$matriz\_de\_rotación = \begin{bmatrix} U(0,2) & -U(1,1) \\ -U(1,1) & U(2,0) \end{bmatrix}$$

Y a partir de estas componentes se puede obtener el ángulo de rotación de la figura a alrededor de su centro de masas. El ángulo de rotación φ se define como el ángulo entre el eje

de abscisas y el eje alrededor del cual el objeto puede girar con mínimo esfuerzo (eje principal). El ángulo de rotación se calcula de la siguiente forma:

$$\varphi = \frac{1}{2}\arctan\frac{b}{a-c}$$

siendo *a*, *b* y *c* los momentos de segundo orden del *blob* y que se calculan de la siguiente forma

$$a = U(2,0) = \sum_{i=0}^{anchura-1}\sum_{j=0}^{altura-1}(x_{ij}-\bar{x})^2 g(i,j)$$

$$b = U(1,1) = 2\sum_{i=0}^{anchura-1}\sum_{j=0}^{altura-1}(x_{ij}-\bar{x})(y_{ij}-\bar{y})g(i,j)$$

$$c = U(0,2) = \sum_{i=0}^{anchura-1}\sum_{j=0}^{altura-1}(y_{ij}-\bar{y})^2 g(i,j)$$

Siendo $\bar{x}$, $\bar{y}$ el centro de gravedad del *blob* y x_{ij}, y_{ij} las coordenadas de los píxeles que forman el *blob* y *g(i,j)* el color del pixel situado en la coordenadas *i,j*. El centro de gravedad es el punto de un objeto que tiene la misma cantidad de objeto en cualquier dirección y sus coordenadas se calculan de la siguiente forma

$$\bar{x} = \frac{\sum_{i=0}^{anchura-1}\sum_{j=0}^{altura-1} j * g(i,j)}{area}$$

$$\bar{y} = \frac{\sum_{i=0}^{anchura-1}\sum_{j=0}^{altura-1} i * g(i,j)}{area}$$

donde g(i,j) representa el color del pixel situado en las coordenadas i,j. Como resultado del cálculo de la orientación se obtiene el ángulo en radianes que el objeto forma con el eje de abscisas, el cual es enviado a través de TUIO al juego.

3.5 Cálculo del contorno

En este apartado se explica en detalle el proceso seguido para el cálculo de los contornos de los objetos reconocidos.

El cálculo del contorno de una imagen se divide en dos fases, la primera consiste en extraer los píxeles que forman parte del borde de la imagen, para luego utilizarlos en la segunda fase de cálculo de los segmentos que forman el contorno. Tras estudiar varios algoritmos para estas tareas (detallado en el Anexo C), finalmente se optó por una mezcla de dos algoritmos, el algoritmo de Moore [Oc, web] para la extracción de los píxeles y una simplificación del algoritmo de cadena [De01] para la obtención del contorno a través de los píxeles calculados en el algoritmo anterior.

El algoritmo de Moore se basa en la extracción de los píxeles que forman el contorno ayudándose de la propiedad de los 8 vecinos (figura 21).

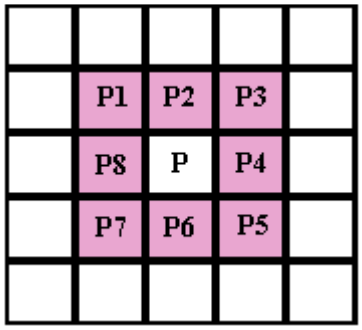


Figura 21 Relación 8-vecinos.

Esta propiedad consiste en lo siguiente: un pixel P_x es un componente 8-vecino de un pixel dado P si comparten o bien una arista o un vértice. Por tanto un conjunto S de píxeles estará 8 conectado, si para cada par de píxeles P y P_x, existe una secuencia de píxeles P…P_x que cumplen que cada 2 píxeles que son adyacentes, se cumple la propiedad de que son 8-vecinos. En la figura 22 se puede ver un ejemplo de un conjunto de píxeles negros que cumplen la propiedad de 8-vecinos y en la figura 23 otro conjunto que no.

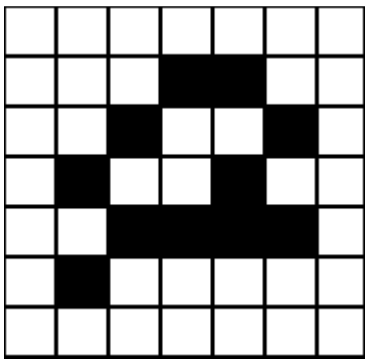


Figura 22 Conjunto que cumple la 8 vecindad. Todos los píxeles comparten un vértice o una arista.

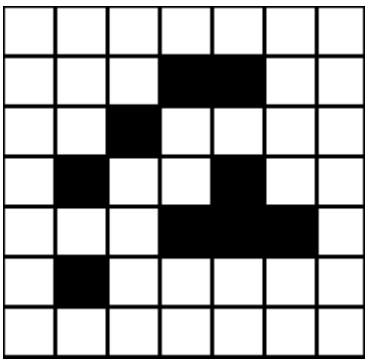


Figura 23 Conjunto que no cumple la 8 vecindad. Hay píxeles que no comparten vértices ni aristas.

La idea del algoritmo consiste en ir recorriendo la imagen en orden contrario al de las agujas del reloj (se pude hacer también al revés) e ir almacenando los píxeles de color blanco del objeto que se va reconociendo. Para ello el algoritmo pasa primero la imagen del objeto reconocido a una matriz, en la cual se guarda los píxeles que forman la imagen, todos ellos blancos o negros. A continuación la matriz se recorre por columnas y de forma ascendente en busca del primer pixel blanco (en Reactivision los elementos reconocidos son blancos y el fondo negro). Una vez encontrado, se marca este pixel como pixel de comienzo y se guarda en un vector, en el que se almacenaran los píxeles que forman el contorno. Después se vuelve al pixel por el que se ha pasado justo antes de encontrar el de comienzo y se recorre los 8 vecinos de este en el orden que se haya establecido hasta encontrar un vecino que sea blanco. Una vez hallado se comprueba que no ha sido ya guardado y se almacena en el vector de píxeles de contorno. A partir del pixel detectado se vuelve a repetir los pasos anteriores. El algoritmo finaliza cuando se haya vuelto a localizar el pixel de comienzo. Un problema que presenta este algoritmo es que se puede dar el caso de volver al pixel de comienzo y que no se hayan recorrido todos los píxeles del contorno. En las figuras 24, 25 y 26 se puede ver un ejemplo de este caso.

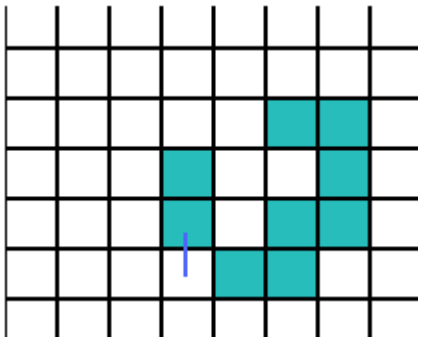


Figura 24 Se encuentra el pixel de comienzo

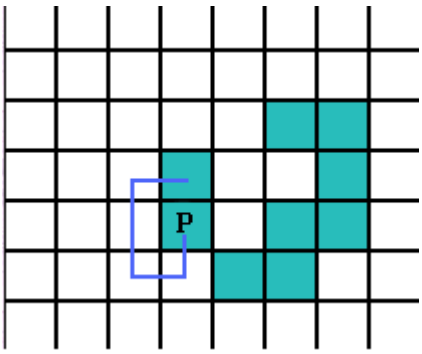


Figura 25 Se recorre los 8 vecinos en busca del siguiente pixel

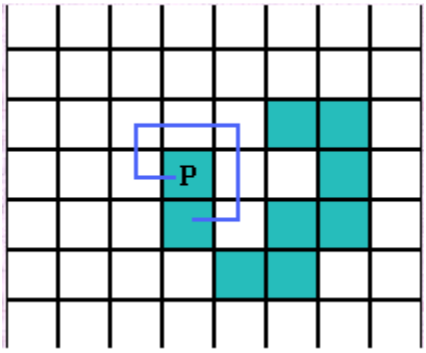


Figura 26 Se busca el siguiente pixel, pero volvemos al pixel de comienzo sin haber recorrido todos los píxeles.

Para solucionar este problema, se cambió el criterio de finalización del algoritmo. Para que el algoritmo pudiese dar por concluido el proceso de búsqueda de píxeles, debía pasar por el pixel de comienzo al menos n veces. Si realizamos de esta manera la búsqueda de los píxeles en el ejemplo anterior, se comprueba como ahora sí que recorre todos los píxeles del contorno. Finalmente se estableció que se pasara por el pixel de comienzo 3 veces, se comprobó que para el tipo de objetos que tratábamos y para su tamaño, esta condición de parada era suficiente.

Otro de los problemas que se detectaron fue que este algoritmo estaba pensado para imágenes sin ruido. En el caso de este proyecto al ser imágenes capturadas en tiempo real, estas suelen poseer ruido de cámara y por tanto dificultan e impiden la correcta búsqueda del pixel de comienzo. Muchas veces el algoritmo detecta un pixel de comienzo que no pertenece al objeto sino que es debido al ruido, por lo que se generan contornos erróneos. En la figura 27 se muestra un ejemplo de una detección que produciría un contorno erróneo.

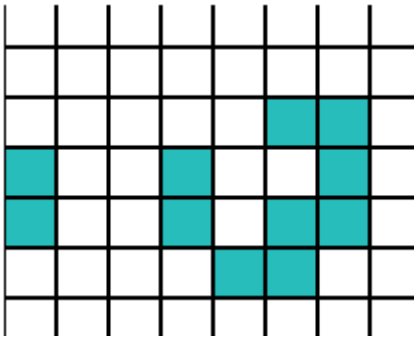


Figura 27 Contorno con ruido, y que por tanto no cumple la 8-vecindad.

Para solucionar este problema se recurrió al estudio de varias técnicas de eliminación de ruido: filtro de moda, filtro de media, filtro de máximos, filtro de mínimo y filtro basado en conectividad entre áreas. Tras analizarlos se determinó que los más efectivos para las imágenes que se tratan en Reactivision, eran el de media y el de conectividad, ya que estos puede adaptarse a imágenes en blanco y negro, mientras que los demás son más efectivos para imágenes en escala de grises.

Primero se realiza un filtro de media a la imagen original que consiste en que para cada pixel se suma su valor y el de sus 8 vecinos, después si el resultado es superior a 0.5, se considera que el pixel tratado es blanco, en caso contrario se almacena como negro. El valor con el que se compara el resultado se puede variar en función de la concentración de ruido en la imagen. En este caso se comprobó que el valor 0.5 era el óptimo, ya que si era menor eliminaba poco ruido y si era mayor se eliminaba parte del objeto reconocido.

Como la eliminación de ruido a través del filtro de media no era completa, se desarrolló otra fase de limpieza basada en un algoritmo de conectividad de píxeles. Este algoritmo consiste en etiquetar todas las áreas blancas que formen parte de la imagen y una vez obtenidas contar el número de píxeles que las forman. Como el ruido normalmente está formado por una pequeña cantidad de píxeles, se procede a convertir a negro todas las áreas cuyo número de píxeles sea inferior a 10 píxeles. Se comprobó que estas dos fases de limpieza eliminaban por completo el ruido de cámara de la imagen.

Una vez obtenidos todos los píxeles que forman el contorno, es necesario ordenar empezando por el pixel de comienzo, el criterio de ordenación en este caso fue el sentido horario. De esta manera es posible asegurar que los segmentos que formen el contorno seguirán el orden correcto y no se cruzaran entre sí. En un principio este paso no se realizaba y se detectó que los contornos generados eran erróneos, en ellos aparecían líneas que se cruzaban con otras. El cruce de líneas era debido a que el algoritmo de Moore no garantiza que los píxeles sean encontrados en el orden correcto y por tanto los vértices generados tampoco lo estén.

Una vez ordenados todos los píxeles del contorno se procede a aplicar una simplificación del algoritmo de cadena. El objetivo de este algoritmo consiste en simplificar el contorno que forma del objeto a través de líneas rectas. Para ello se debe reducir la matriz que contiene los píxeles obtenidos y calcular así los nodos que unirán los segmentos. Se debe escoger una densidad (número de píxeles) por la cual se divide la matriz y así queda simplificada en secciones. En el caso de este proyecto se ha escogió una densidad de 5

píxeles. De esta manera la matriz queda dividida a la quinta parte y el contorno será mucho más sencillo. En el caso de este proyecto esta densidad es suficiente ya que no necesitamos contornos muy precisos, es decir solo necesitamos contorno aproximados de los objetos, si se necesitan contornos muy exactos tan solo hay que disminuir la densidad elegida. Una vez dividida la matriz, se consideraran nodos del contorno solo aquellos que estén más cercanos a los píxeles de contorno y en los que en su coordenada 'x' o 'y' hay un pixel de los obtenidos en el paso anterior. En la figura 28 se muestra un ejemplo de los vértices que finalmente formarán parte del contorno.

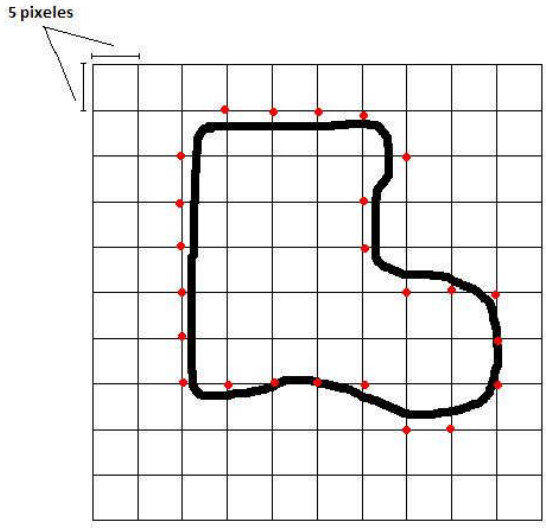
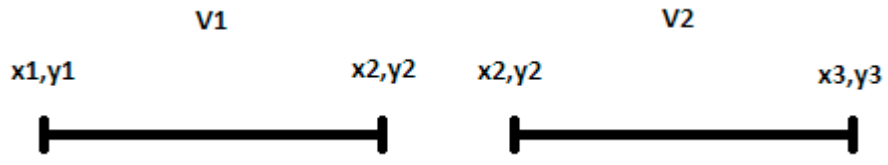


Figura 28 En rojo se muestran los vértices que formarán el contorno. En negro se observa ver la silueta del objeto sin simplificar.

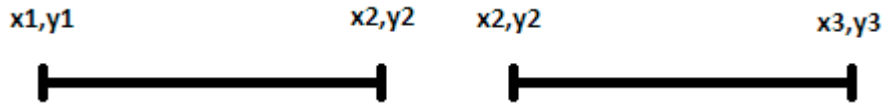
Por último se procede a obtener los segmentos que forman el contorno: uniendo cada dos vértices mediante una línea. Sin embargo muchos de estos segmentos serán paralelos y consecutivos entre sí y podrán forman una única línea, por ello se procede al simplificado de estos en una misma línea de manera que la estructura final sea mucho más sencilla. Para simplificar los segmentos se cogen los tres primeros vértices, con el primero y el segundo se forma un vector y con el segundo y el tercero otro. De esta manera se calcula el producto escalar de estos y se obtiene el ángulo que forman. Si este ángulo es 0 indica que los segmentos son paralelos y se pueden simplificar en una sola línea, después se seguirá comprobando con los demás vértices. Si por el contrario no lo son, se almacena el vector formado por los dos primeros vértices y se pasará a comprobar los siguientes.



$$\overline{v_1 v_2} = |\overline{v_1}||\overline{v_2}| * \cos\alpha$$

Si $\alpha = 0$ →

Sino



En la figura 29 se puede ver un ejemplo de simplificación.

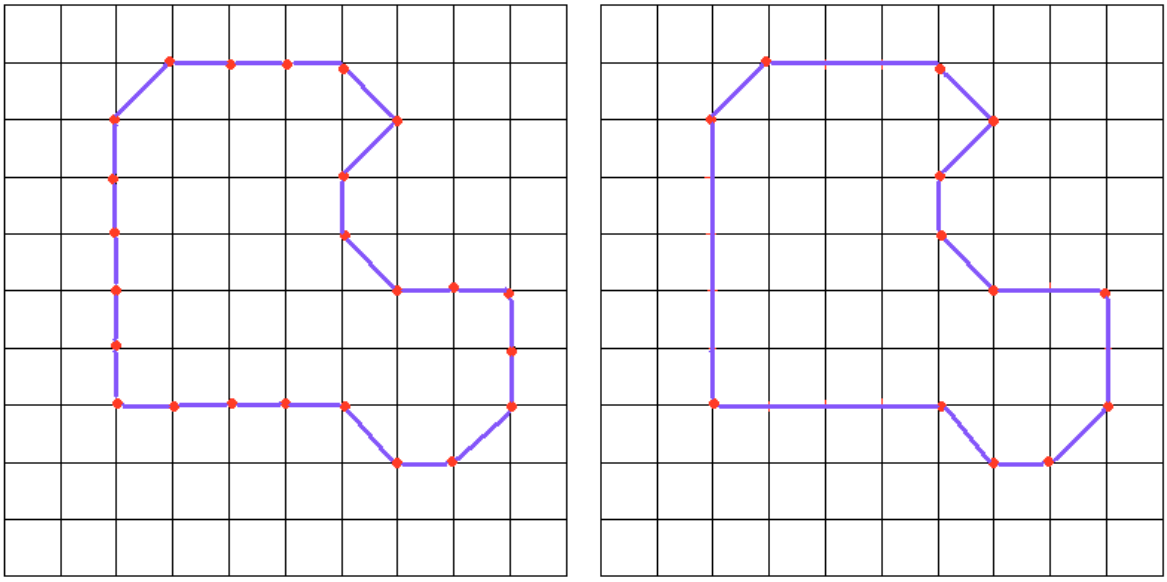


Figura 29 Simplificación de los segmentos que forman el contorno.

Finalmente se tienen ya almacenados los vértices y segmentos del contorno construido y sólo queda enviar esta información a los juegos que o vayan a necesitar.

3.6 Resultados obtenidos

Como resultado del diseño y la implementación de los algoritmos citados se ha superado la limitación que Reactivision tenía frente a la detección de objetos sin fiducial asociado. Se han conseguido extraer características útiles de este tipo de juguetes, que pueden ser empleadas para el diseño de juegos con nuevas funcionalidades.

El cálculo del área permite obtener de forma fiable el tamaño del objeto situado en la superficie del *tabletop*, ya que hasta ahora solo se tenía un cálculo aproximado a partir de su altura y anchura. En la figura 30 se puede ver cómo según el tamaño de los objetos se obtienen unos valores de área u otros.

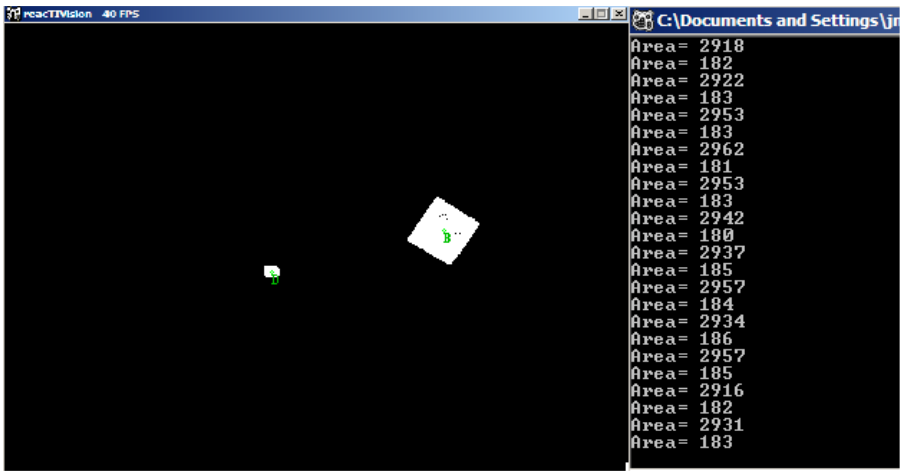


Figura 30 Objetos reconocidos y sus aéreas.

Esta característica se ha empleado en diferentes juegos como por ejemplo en el juego de pintar. El cálculo del área de los objetos permite utilizar pinceles de diferentes grosores consiguiendo así que los trazos que se produzcan sean en función de este grosor o de la presión que el niño ejerza sobre estos, consiguiendo así aumentar las capacidades del juego. Sin esta característica se conseguirían siempre trazos del mismo grosor y se perdería la posibilidad de utilizar pinceles de diferentes tamaños. En la figura 31 se puede ver un ejemplo del juego de pintar, en el que se observan trazos de diferentes tamaños en función del pincel utilizado.

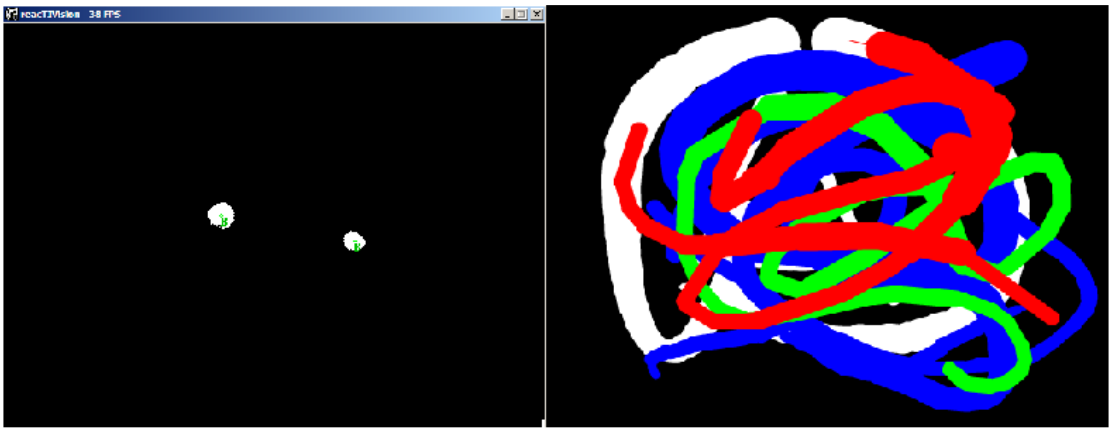


Figura 31 Captura de Reactivision en la que se pueden ver los grosores de los pinceles detectados y los diferentes trazos producidos en el juego de pintar.

Por otra parte la extracción de la orientación de los objetos permite saber hacia donde están apuntados los juguetes, simplemente a través de su eje principal. Notar que los ángulos siempre están comprendidos entre 0º y 180º ya que al utilizar el eje de menor inercia el cual no tiene dirección especifica. El cálculo del eje no distingue entre la parte de delante y la de detrás del objeto por lo que no se puede calcular una orientación con un rango de 360º. Esta característica permite por ejemplo calcular la dirección de un juguete sin tener que añadirle ningún tipo de fiducial. En la figura 32 se observa cómo se calcula de una forma precisa el ángulo de orientación con respecto al eje de abscisas ‘x’.

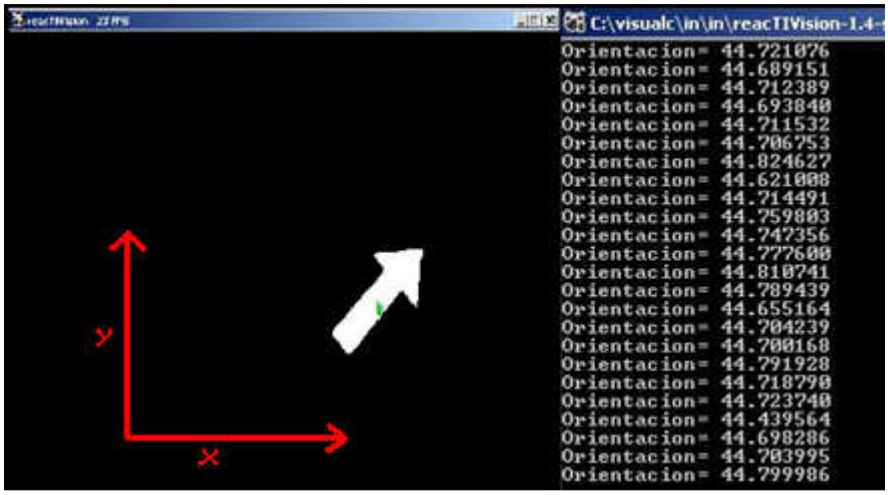


Figura 32 Objeto identificado en Reactivision y valores de su orientación.

Por otra parte en la figura 33 se observa diferentes objetos sin fiducial, reconocidos por Reactivision y sus contornos calculados a partir de los algoritmos anteriormente explicados. Estos contornos permiten establecer la forma aproximada de los objetos situados en el *tabletop* permitiendo usar cualquier tipo de juguete sin la necesidad de su modificación.

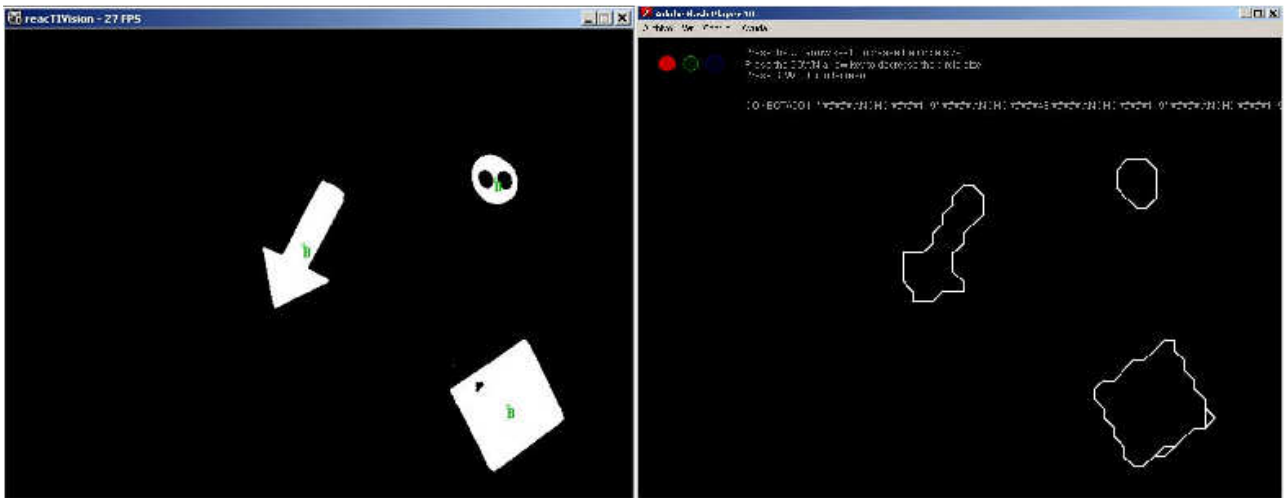
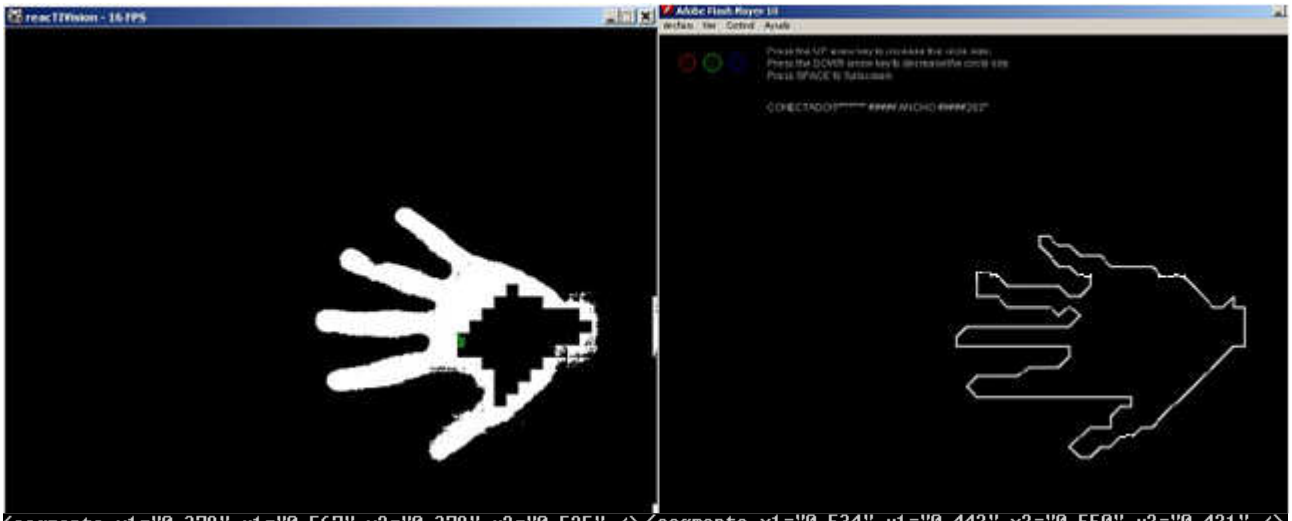


Figura 33 Ejemplo de objetos y el cálculo de sus contornos.

Como se puede ver a continuación también se obtiene buenos resultados con objetos de formas más complejas. En la figura 34 se puede observar el resultado del cálculo del contorno de una mano, así como su representación a través de sus segmentos. Se puede ver como con unos pocos segmentos, se ha conseguido representar de forma bastante precisa el contorno de cualquier objeto. De esta manera los niños pueden utilizar cualquier objeto a su alcance, lo que hace mucho más dinámicos los juegos que usen esta característica.



```
<segmento x1="0.378" y1="0.567" x2="0.378" y2="0.525" /> <segmento x1="0.534" y1="0.442" x2="0.550" y2="0.421" />
<segmento x1="0.378" y1="0.525" x2="0.394" y2="0.504" /> <segmento x1="0.550" y1="0.421" x2="0.550" y2="0.442" />
<segmento x1="0.394" y1="0.504" x2="0.394" y2="0.442" /> <segmento x1="0.550" y1="0.442" x2="0.534" y2="0.463" />
<segmento x1="0.394" y1="0.442" x2="0.409" y2="0.421" /> <segmento x1="0.534" y1="0.463" x2="0.519" y2="0.483" />
<segmento x1="0.409" y1="0.421" x2="0.409" y2="0.400" /> <segmento x1="0.519" y1="0.483" x2="0.503" y2="0.483" />
<segmento x1="0.409" y1="0.400" x2="0.425" y2="0.400" /> <segmento x1="0.503" y1="0.483" x2="0.503" y2="0.504" />
<segmento x1="0.425" y1="0.400" x2="0.409" y2="0.421" /> <segmento x1="0.503" y1="0.504" x2="0.487" y2="0.525" />
<segmento x1="0.409" y1="0.421" x2="0.409" y2="0.463" /> <segmento x1="0.487" y1="0.525" x2="0.472" y2="0.546" />
<segmento x1="0.409" y1="0.463" x2="0.425" y2="0.483" /> <segmento x1="0.472" y1="0.546" x2="0.487" y2="0.567" />
<segmento x1="0.425" y1="0.483" x2="0.425" y2="0.463" /> <segmento x1="0.487" y1="0.567" x2="0.519" y2="0.567" />
<segmento x1="0.425" y1="0.463" x2="0.441" y2="0.442" /> <segmento x1="0.519" y1="0.567" x2="0.534" y2="0.546" />
<segmento x1="0.441" y1="0.442" x2="0.441" y2="0.400" /> <segmento x1="0.534" y1="0.546" x2="0.550" y2="0.567" />
<segmento x1="0.441" y1="0.400" x2="0.456" y2="0.379" /> <segmento x1="0.550" y1="0.567" x2="0.534" y2="0.567" />
<segmento x1="0.456" y1="0.379" x2="0.456" y2="0.358" /> <segmento x1="0.534" y1="0.567" x2="0.519" y2="0.588" />
<segmento x1="0.456" y1="0.358" x2="0.472" y2="0.358" /> <segmento x1="0.519" y1="0.588" x2="0.503" y2="0.588" />
<segmento x1="0.472" y1="0.358" x2="0.472" y2="0.379" /> <segmento x1="0.503" y1="0.588" x2="0.487" y2="0.608" />
<segmento x1="0.472" y1="0.379" x2="0.456" y2="0.400" /> <segmento x1="0.487" y1="0.608" x2="0.472" y2="0.608" />
<segmento x1="0.456" y1="0.400" x2="0.456" y2="0.442" /> <segmento x1="0.472" y1="0.608" x2="0.456" y2="0.629" />
<segmento x1="0.456" y1="0.442" x2="0.441" y2="0.463" /> <segmento x1="0.456" y1="0.629" x2="0.441" y2="0.629" />
<segmento x1="0.441" y1="0.463" x2="0.441" y2="0.483" /> <segmento x1="0.441" y1="0.629" x2="0.425" y2="0.608" />
<segmento x1="0.441" y1="0.483" x2="0.487" y2="0.483" /> <segmento x1="0.425" y1="0.608" x2="0.394" y2="0.608" />
<segmento x1="0.487" y1="0.483" x2="0.503" y2="0.463" /> <segmento x1="0.394" y1="0.608" x2="0.378" y2="0.588" />
<segmento x1="0.503" y1="0.463" x2="0.519" y2="0.442" /> <segmento x1="0.378" y1="0.588" x2="0.378" y2="0.567" />
<segmento x1="0.519" y1="0.442" x2="0.534" y2="0.442" />
```

Figura 34 Mano capturada y reconstrucción de su contorno así como su representación en formato XML.

Capítulo 4. Protocolo de comunicación.

La comunicación entre el *framework* Reactivision y los juegos diseñados en Flash se realiza a través del protocolo TUIO. Este protocolo (ver anexo A) fue diseñado en un principio para el envío de las características específicas de objetos con fiducial, que recogen los siguientes parámetros: identificador del fiducial, posición y orientación. De esta manera los paquetes UDP que este protocolo utilizaba limitaban el número de datos a enviar así como la estructura que éstos debían tener. La implementación de nuevas funcionalidades y la necesidad de mejora de otras, hizo necesaria la creación de un nuevo protocolo de comunicación que permitiese el envío de grandes volúmenes de datos, así como de información de distinto tipo con estructuras complejas.

En este capítulo se detalla el diseño e implementación del nuevo protocolo de comunicación así como los diferentes usos que se le ha dado. En el anexo C se explican en detalle otras alternativas de formas de comunicaciones estudiadas y finalmente descartadas.

4.1 Definición del protocolo

Tras estudiar otras alternativas (anexo C), finalmente se decidió implementar un protocolo basado en la comunicación de información codificada en ficheros XML a través de sockets TCP. Estas dos tecnologías fueron escogidas por sus numerosas ventajas. El formato XML es un estándar muy utilizado y además permite el uso de multitud de lenguajes para su tratamiento. Esto asegura la posibilidad de diseñar los juegos en varios lenguajes y no limita al uso de un solo lenguaje, lo que en un futuro podría llegar a ser un gran problema si se quisiera extender a otros *frameworks*. Además permite codificar y leer información con estructuras muy diversas de una manera sencilla y por tanto no condiciona al uso de datos simples. Por otra parte es un lenguaje fácil de comprender, ya que puede usar etiquetas creadas a medida, por lo que facilita la compresión por parte de usuarios inexpertos o sin nociones de lenguajes de programación.

Por su parte la comunicación a través de socket asegura la transmisión de toda la información sin errores ni omisiones y a su vez garantiza que dato enviado llegará a su destino en el mismo orden en que se ha transmitido. Estas propiedades son muy importantes para garantizar la corrección de los programas que tratan la información. En los juegos no se puede permitir la pérdida o la corrupción de información porque de esta manera se estaría ralentizando el juego eliminando por completo la finalidad de estos, es decir divertir. Por otra parte al igual que XML el uso de sockets, permite el uso de múltiples lenguajes.

Una vez teniendo claras las ventajas de las dos tecnologías, se procedió a la búsqueda de información sobre la implementación de sockets en los dos lenguajes empleados en este proyecto: C++ utilizado para la implementación de Reactivision y el lenguaje ActionScript mediante el que se desarrollan los juegos para NIKvision. Para C++ se encontró la librería '*winsock*', biblioteca dinámica de funciones DLL para Windows que incluye soporte para envío y recepción de paquetes de datos a través de sockets TCP. Para el uso de sockets en ActionScript se encontró la librería '*flash.net.socket*', que permite emplear sockets como método de comunicación. Sin embargo esta sólo implementa la parte del cliente, por este

motivo Reactivision tenía que actuar siempre como servidor, por lo que se decidió crear una clase `XMLSocket`, que se encargase de la comunicación del servidor.

La clase `XMLSocket` en principio debía contener una función que se encargaría de la conexión inicial con los juegos y otra que gestionase del envío de la información recopilada. Una vez implementadas estas dos funciones, se procedió a realizar las primeras pruebas de conexión. Estas pruebas obligaron a modificar la clase implementada, ya que se observó que la conexión se establecía, pero el envío de la información fallaba. Se detectó que el fallo se encontraba en la política de seguridad que Flash obliga a implementar. Esta política exige a que antes de realizar cualquier envío de datos entre el servidor y el cliente, el servidor deba enviar una cadena XML al cliente en la que se detalle la política de conexión entre ambos. Esta política, consiste en indicar al cliente la dirección IP y el puerto del servidor desde el que se va a realizar la conexión.

Tras modificar la función de conexión para que realizase el envío de la política de seguridad se volvió a repetir la prueba anterior. El cliente tras recibir la política de seguridad daba un error, esta vez sí se establecía la conexión socket entre cliente y servidor, pero el cliente seguía sin recibir datos, esto era debido a la necesidad de dos servidores: uno para el envío de la política y otro para la comunicación de datos. Se implementaron por tanto dos servidores: el primero se limita a aceptar la conexión por parte del cliente y a enviar la cadena XML que contiene la IP y puerto del servidor de datos y el segundo se encarga de la comunicación de los datos con el cliente. De esta manera el protocolo queda establecido de la siguiente forma: el servidor de datos queda esperando la conexión de un cliente, cuando esta conexión se produce envía al cliente la cadena con los datos del servidor de datos. Una vez recibida, el cliente cierra la conexión socket e intenta conectar con el servidor de datos, este último acepta la conexión y ya que puede comenzar enviar la información.

4.2 Usos del protocolo

A continuación se detallan las dos funciones que se han implementado haciendo uso del nuevo protocolo de comunicación: envío de un bitmap y envío de la información extraída de objetos sin fiducial. Este protocolo sin embargo puede extenderse para otras muchas aplicaciones, tan solo modificando los ficheros XML.

4.2.1 Envío de un bitmap

El procesamiento de la información capturada por medio de Reactivision es un proceso costoso en tiempo. La velocidad de procesamiento es crítica por ejemplo en el juego de pintar, en el que se requiere enviar dibujos situados en la superficie del *tabletop*. El software de detección reconocía la hoja de papel que contenía el dibujo a través de fiducial impreso en ella. Después la imagen capturada era procesada para extraer el dibujo que esta contenía y a continuación la imagen era convertida en un fichero *bmp* y guardada en una carpeta de destino local.

Una vez guardada la imagen, se informaba al juego del reconocimiento del dibujo mediante un mensaje TUIO. El juego accedía a la carpeta en la que se había guardado la

imagen para poder leerla y trabajar con ella. Por último éste borraba la imagen para que Reactivision pudiese enviar otro dibujo sin borrar el anterior. Este modo de compartir información era lento y ralentizaba el juego, ya que Reactivision quedaba bloqueado hasta que Flash no terminaba de gestionar el dibujo. El desarrollo del un nuevo protocolo de comunicación permite enviar la imagen a través de una cadena XML indicando el valor de cada uno de los píxeles que la forman, evitando así el problema del bloqueo.

Por otra parte, para evitar que los dibujos mostrados por pantalla aparecieran ensuciados, se procede a limpiar la imagen y se elimina en la mayor medida las machas producidas por el ruido de cámara. Para lograr dicho objetivo se decide aplicar un filtro de media que permite suavizar la imagen, haciendo una media del valor de los píxeles. Se optó por utilizar un filtro similar al presentado en el capítulo 3.

Una vez limpia la imagen se procede a enviarla a través de la cadena XML. Para reducir aún más el tiempo de envío, se buscaron otras estrategias: en lugar de enviar la imagen por completo, se decidió enviar una de cada dos líneas, de esta manera se reducía a la mitad el volumen de datos. Se comprobó que el resultado era lo suficientemente bueno para el nivel de detalle que se deseaba. La estructura XML establecida para el envío del dibujo es la siguiente:

```
<imagen>
      <pixel valor=”0”>
      <pixel valor=”1”>
      <pixel valor=”1”>
      <...>
</imagen>
```

Una etiqueta `pixel` para cada uno de los píxeles enviados donde `valor` indica si el pixel es negro o blanco. Tras implementar el envío y realizar pruebas de su funcionamiento, se comprobó que los XML generados eran demasiado grandes, lo que provocaba problemas de almacenamiento y lentitud al almacenar el XML en una cadena de texto, para su posterior envío. El problema residía en su totalidad en la forma de representación de la imagen, como ya se ha comentado se enviaba la imagen pixel a pixel, indicado el color de cada pixel. Como solución a este problema se llevó a cabo una compresión de la información. El método de compresión elegido, por ser un método sencillo de aplicar, fue el método RLE (Run Length Encoding). Este método consiste en la compresión de datos de manera que secuencias de datos consecutivos con el mismo valor son almacenadas como un único valor más su recuento. A continuación se muestra un ejemplo de una cadena comprimida con el método RLE:

Información:	AAAAABBAAABBBBA
Información comprimida:	5A2B3A3B1A

Este método es útil si las secuencias de datos iguales son largas, sino puede llegar a generar una compresión con un tamaño mayor a la información original, ya que se estarían enviando continuamente secuencias intercaladas de un solo pixel con valores distintos. En el caso del dibujo se ajusta perfectamente a los requerimientos del algoritmo, ya que la mayoría de píxeles van a ser negros por el fondo y solo de vez en cuando habrá secuencias cortas de píxeles blancos debidas al dibujo. Una comprimida la información, se modificó la estructura de la cadena XML para que soportase la nueva estructura de los datos. La estructura finalmente establecida es la siguiente:

```
<dibujo ancho=”” alto=””>
      <pixel cantidad=”” valor=””>
```

```
      <...>
      <pixel cantidad=”” valor=””>
</ dibujo >
```

donde `cantidad` indica el número de píxeles consecutivos con el mismo valor y `valor` indica el color del pixel. También se han añadido la altura y anchura del dibujo, de manera que se facilite el posterior tratamiento de la imagen por parte de los juegos. A continuación se puede ver un ejemplo real de la cadena XML con la información referente un dibujo (fig 35).

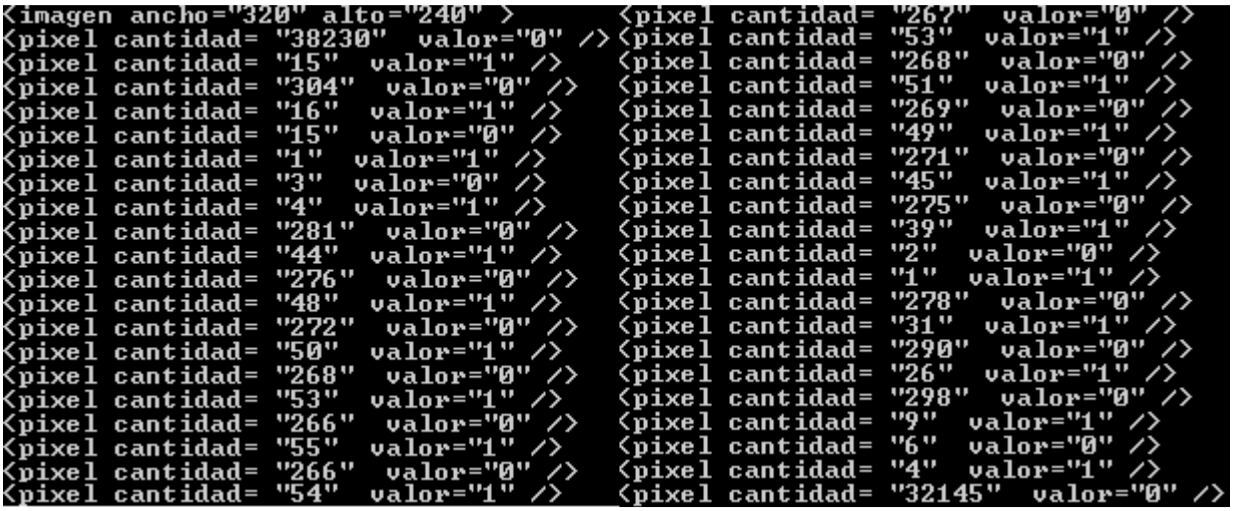


Figura 35 XML que representa un dibujo detecta por Reactivision

El envío de un dibujo se realiza cada vez que Reactivision reconoce el fiducial asociado al papel y el identificador de éste corresponda con el identificador añadido en el fichero de configuración (capítulo 5).

4.2.2 Envío de información de objetos sin fiducial

Como ya se ha explicado este nuevo protocolo de envío también fue diseñado para agilizar el paso de información entre Reactivision y los juegos en Flash. Por lo tanto puede utilizarse en juegos que requieran de información con estructuras muy diferentes y grandes volúmenes de datos. Este avance ha permitido el envío de información relativa a los objetos sin fiducial, que en ese momento estuviesen sobre la mesa.

Los datos enviados del objeto sin fiducial son: coordenadas de las esquinas superior izquierda e inferior derecha del objeto detectado, su contorno, orientación y área, así como una imagen de toda la superficie del *tabletop*. La nueva estructura establecida para el envío de esta información es la siguiente:

```
<blob ancho=”” alto=”” x=”” y=”” area=”” orientación=””>
      <segmento x1=”” y1=”” x2=”” y2=””/>
      <...>
      <segmento x1=”” y1=”” x2=”” y2=””/>
</blob>
...
<blob ancho=”” alto=”” x=”” y=”” area=”” orientación=””>
      <segmento x1=”” y1=”” x2=”” y2=””/>
```

```
        <...>
        <segmento x1="" y1="" x2="" y2=""/>
    </blob>
    <imagen ancho="" alto="">
        <pixel cantidad="" valor="">
        <...>
        <pixel cantidad="" valor="">
    </imagen>
```

donde cada etiqueta `blob` representa un juguete reconocido sin fiducial junto con sus características, las etiquetas `segmentos` representan las líneas que forman el contorno calculado del juguete y la etiqueta `imagen` representa la imagen que se envía de la superficie completa del *tabletop*. Esta última información se comprime de la misma manera que el dibujo explicado en el apartado anterior.

4.3 Resultados obtenidos

Como resultado del diseño del nuevo protocolo de comunicación y de su implementación, han sido superadas las limitaciones que el protocolo TUIO presentaba en cuanto al envió de la información obtenida a través Reactivision a los juegos en Flash.

En un primer lugar la utilización del lenguaje XML en el protocolo, permite el envío de datos con estructuras muy diferentes. Al ser un lenguaje de etiquetas, se pueden definir estructuras personalizadas para la información que se desee enviar. De esta manera modificando sólo la estructura del fichero, se asegura que cualquier información pueda ser enviada a los juegos. A su vez al enviarse cadenas de texto, y no paquetes con una longitud determinada, se puede enviar cualquier volumen de información, pudiendo ser cantidades de datos muy grandes. Con el protocolo TUIO nos debíamos limitar a un número muy reducido de datos y además con unas estructuras muy simples (enteros, flotantes, etc).

En la figura 36 se puede ver un ejemplo de los ficheros enviados cuando se reconocen objetos sin fiducial en la superficie del *tabletop*. Se detectan dos objetos, por lo que se envía un fichero XML que contiene tanto las coordenadas de los objetos, su área, orientación, así como una lista de los segmentos que forman el contorno de ambos objetos.

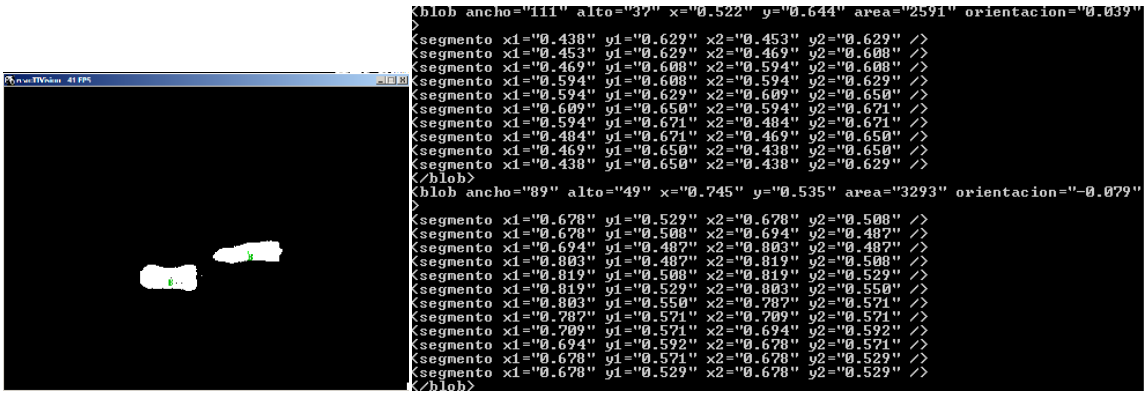


Figura 36 Objetos sin fiducial reconocidos por Reactivision y XML con la información correspondiente a dos objetos

A su vez gracias a que tanto los sockets como el lenguaje XML son multiplataforma, se asegura que sea cual sea el lenguaje en el que estén desarrollados los juegos, no habrá problema de compatibilidad con el protocolo de comunicación diseñado.

Por otro lado la forma de envío de los datos y al algoritmo de compresión mejoran el envío de los dibujos en el juego de pintar. El principal problema que existía con respecto a la implementación anterior (basadas en ficheros .bmp) era la complejidad en cuanto a los pasos a implementar para la trasmisión del dibujo:

- Una vez reconocido el dibujo, Reactivision generaba y guardaba el fichero .bmp en una carpeta predefinida y enviaba un mensaje TUIO al juego Flash.
- Por la parte de Flash cuando se recibía dicho mensaje se accedía y leía el fichero .bmp y se mostraba en pantalla. Tras ello se procedía a borrar el fichero, ya que Reactivision no podía generar otro fichero en dicha carpeta mientras todavía permaneciese el anterior, para así evitar la sobre escritura de este mientras Flash no hubiese acabado de leerlo.

Este proceso además de complejo de implementar, provocaba bloqueos de espera en Reactivision dependientes del funcionamiento de Flash. Con el nuevo protocolo el proceso es mucho más sencillo:

- Reactivision envía a través de un socket los datos del dibujo y el juego Flash los recibe.

Con esta situación no se producen situaciones de conflicto en los que Reactivision quede bloqueado por la ejecución de Flash. En la figura 37 se muestra un ejemplo de la cadena XML enviado al reconocer el fiducial asociado a un dibujo. En se puede observar ver la comprensión utilizada para el envío de todos los píxeles que forman la imagen.

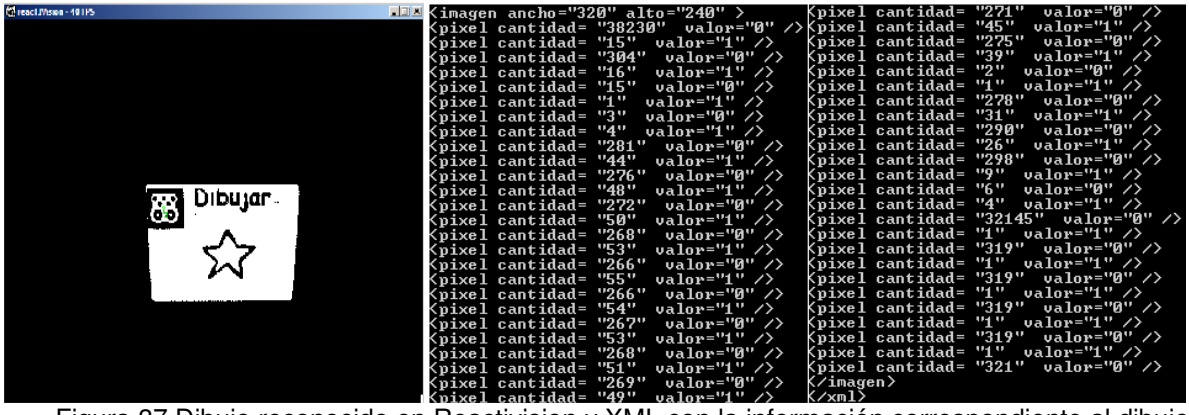


Figura 37 Dibujo reconocido en Reactivision y XML con la información correspondiente al dibujo

Capítulo 5. Interfaz de usuario.

Una de los puntos más importantes en los programas informáticos es la interfaz de usuario. Una buena interfaz permite el manejo del sistema de una forma mucho más sencilla y facilita la tarea a los usuarios. Debido a las nuevas funcionalidades realizadas en este proyecto se ha modificado la interfaz de usuario de manera que integre las nuevas funcionalidades y parámetros utilizados. A su vez se han modificado y mejorado los ficheros de configuración de Reactivision incluyendo la posibilidad de añadir los nuevos parámetros generados así como otros que no estaban previamente incluidos.

En este capítulo se detallan las modificaciones realizadas en el interfaz de usuario y las mejoras incluidas en los ficheros de configuración.

5.1 Interfaz de usuario

La configuración y visualización de las opciones se desarrolló de forma similar al resto de configuraciones de valores en Reactivision. Todas ellas se basan en la visualización por pantalla de barras en las que se puede observar el valor de la variable a tratar y en algunas ocasiones estos valores se acompañan de gráficos para identificar de manera más clara as modificaciones realizadas. El acceso al menú de interfaz de usuario se realiza mediante teclado y en la ventana de inicio de Reactivision se pueden observar las teclas a pulsar para acceder al menú de cada uno de los elementos configurables. Una vez dentro de cada menú, los valores se modificarán utilizando las flechas 'derecha' e 'izquierda' y la navegación por el menú será a través de las flechas 'arriba' y 'abajo'. En la figura 38 se puede ver la ventana de inicio y las diferentes opciones de configuración.

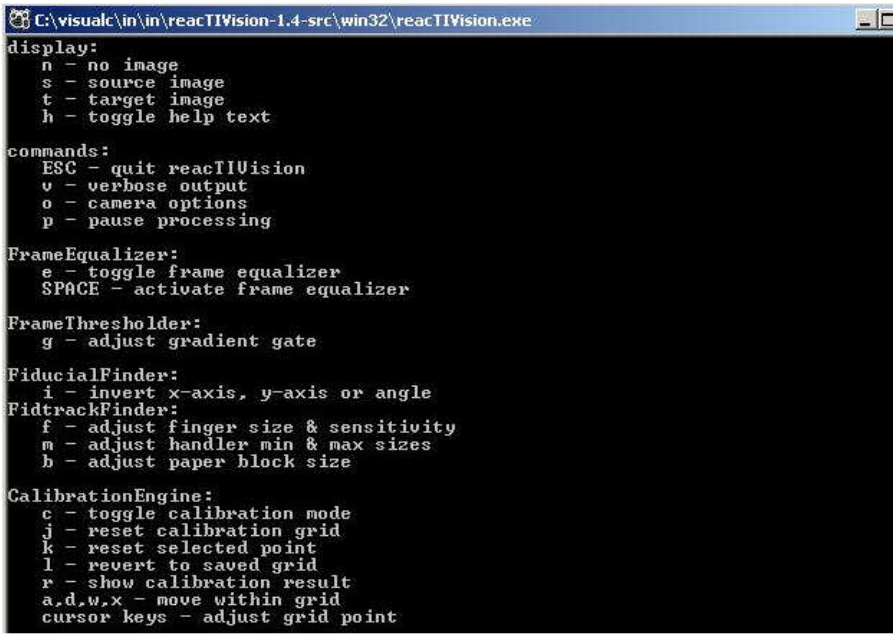


Figura 38 Ventana de inicio de Reactivision

La función para ello modificada ha sido drawGUI, encargada de mostrar y gestionar los cambios en los diferentes parámetros configurables. Para la configuración de los nuevos

fiduciales, se ha añadido la posibilidad de controlar el número de píxeles blancos que estos deben tener por lo tanto como el tamaño de los agujeros también viene en relación con el tamaño del área blanca del fiducial, esta opción permite controlar el tamaño de los fiduciales. Esta nueva opción complementa a la posibilidad de cambiar el tamaño de los *finger* detectados (opción por defecto en Reactivision), ya que los nuevos fiduciales comparten las opciones de tamaño de éstos. Además la posibilidad de regular el área blanca de los fiduciales permite asegurar que estos no son confundidos con ruido de cámara, tal y como se ha explicado en el capítulo 2. El menú para cambiar estos valores se activa pulsando la tecla 'f'. En la figura 39 se puede ver las distintas opciones de configuración de los nuevos fiduciales.

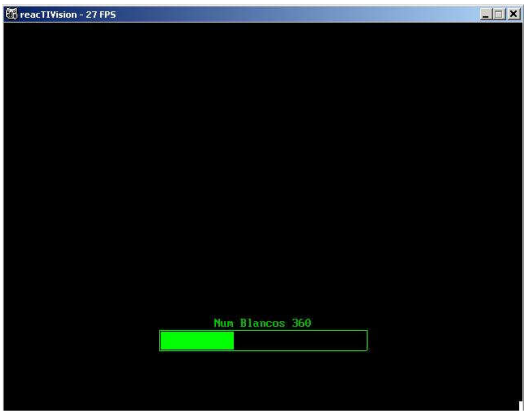


Figura 39 Menú de configuración del número de píxeles blancos de los nuevos fiduciales

La interfaz también se ha modificado para permitir configurar el tamaño de los folios donde los niños pueden dibujar en el juego de pintar, así como el fiducial que éste lleva asociado. Hasta ahora se podía modificar los valores de altura y anchura del papel así como el tamaño del fiducial, sin embargo estos valores eran difíciles de comprender por parte del usuario, ya que estaban expresados en píxeles lo que hacía difícil hacerse a la idea del tamaño real del papel y su fiducial. Para eliminar esta limitación se ha asociado al menú de configuración unos gráficos que representan el tamaño del papel una vez reconocido por Reactivision, de manera que el usuario antes de empezar a jugar puede colocar un papel en el *tabletop* y configurar su tamaño visualizando en todo momento el área capturada. En la figura 40 se puede ver el menú de configuración del papel. Este menú se activa con la pulsación de la letra 'b'.

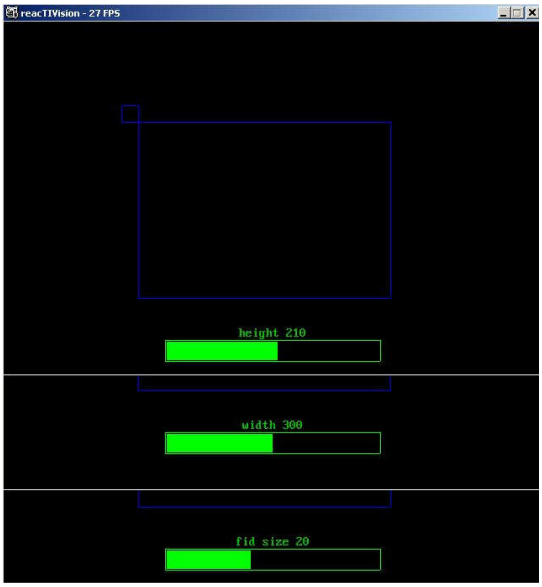


Figura 40 Menú de configuración del papel, para el juego de pintar.

5.2 Ficheros de configuración

Uno de los problemas que presenta Reactivision es la inicialización de los parámetros que se van a utilizar en los juegos: tamaños del fiducial, de los *fingers*, de los objetos sin fiducial, de los folios para dibujar, etc. Cada vez que se quiere jugar, lo primero que hay que hacer es configurar todos los valores de estos parámetros, para el reconocimiento de los juguetes a utilizar. Este requerimiento es repetitivo y poco práctico, por lo que en este proyecto se han modificado los ficheros de configuración que Reactivision usaba por defecto, de manera que todos los parámetros se pudiesen incluir en ellos y no hubiese que configurarlos al inicio de cada juego. Así para cada vez que se quiera jugar, solamente se debe cargar el fichero correspondiente a ese juego y todos los valores de los parámetros serán inicializados automáticamente.

A continuación se detallan las posibilidades de configuración incluidas en los ficheros y su modo de funcionamiento.

Objetos sin fiducial

La siguiente instrucción se añade para configurar los parámetros de tamaño de los juguetes que no llevan asociado un fiducial es:

```
<handler min="20" max="200" />
```

La etiqueta `handler`, indica que se modifica el tamaño de los objetos sin fiducial que reconoce el juego. Los parámetros `min` y `max` son los tamaños mínimo y máximo que los

juguetes pueden tener en el juego. Cualquier juguete sin fiducial con un tamaño fuera de este rango no será reconocido.

Nuevos fiduciales

La siguiente instrucción configura los parámetros de los nuevos fiduciales:

```
<finger size="26" sensitivity="74" num_blancos="520" />
```

`num_blancos` es el número de píxeles blancos que debe tener un fiducial de la nueva colección como mínimo para que sea considerado fiducial de este tipo. Si se omite este parámetro, el valor por defecto será 500. Además se añadió una posibilidad de indicar si iban a ser utilizados fiduciales de este tipo o juguetes sin fiducial

```
<mode object="finger2" />
<mode object="blob" />
```

La etiqueta `mode` indica el modo en el que se trabaja y el parámetro `object` indica si se va a trabajar con fiduciales de la nueva colección o bien objetos sin fiducial. Si este parámetro tiene el valor `finger2`, indica que se van a utilizar fiduciales de la nueva colección, mientras que si tiene el valor `blob`, serán los objetos sin fiducial los reconocidos. Esta opción es excluyente, lo que indica que no pueden utilizarse a la vez ambos tipos de objeto. Si se utilizan fiduciales de la nueva colección, los objetos sin fiducial no serán reconocidos y viceversa. Si se omite esta línea en el fichero de configuración indica que en el juego no se van a utilizar fiduciales de la nueva colección ni objetos sin fiducial, de manera que ninguno de ellos será reconocido aunque sean colocados en el *tabletop*.

Papel de dibujo

Para configurar los parámetros de tamaño del papel que contendrá los dibujos, así como el del fiducial asociado, la instrucción que hay que añadir es:

```
<paper height="210" width="300" fid_size="20" />
```

La etiqueta `paper`, indica que vamos a modificar el tamaño de las hojas de papel en el juego. Los parámetros `height` y `width` indican la altura y anchura de la hoja de papel a utilizar y `fid_size` indica el tamaño del fiducial identificador de la hoja de papel.

Si se omite esta línea en el fichero de configuración indica que no se van a utilizar papeles para dibujo en el juego, de manera que no será reconocido ningún dibujo colocado en el *tabletop*.

Protocolo de comunicación

En el fichero de configuración se ha añadido el parámetro segundos que indica cada cuanto tiempo se envía la cadena XML con la información de los objetos sin fiducial al juego en flash.

```
<tiempo segundos="10" />
```

Si la línea se omite, el programa no envía el fichero XML con la información correspondiente a los juguetes reconocidos.

Así mismo se incluye la posibilidad de poder indicar el número identificador del fiducial asociado a los dibujos en el juego de pintar.

```
<dibujo id="11" />
```

`id` indica el identificador del fiducial que tendrá la hoja del dibujo asociado. Si la línea se omite, el programa asigna por defecto el valor -1 y no será reconocido ningún fiducial como perteneciente al dibujo.

En la figura 41 se muestra un ejemplo de fichero de configuración. Este fichero indica que se reconoce dibujos con un tamaño de 384*165 píxeles y con un fiducial asociado de 50 píxeles. A su vez se reconocen los juguetes sin fiducial de entre 25*25píxeles y 150*150píxeles. Se reconocerá también *fingers* con un tamaño de 32*32píxeles y fiduciales cuya topología este almacenada en el fichero árboles `miniset.trees`. Por último se indica que cada 5 segundo se envía la información de los juguetes sin fiducial a los juegos, mediante el nuevo protocolo de comunicación. También se envía a través de él un fichero con la información del dibujo cada vez que se reconozca el fiducial con identificador 10.

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<reactivision>
    <finger size="32" sensitivity="74"/>
    <image display="dest" equalize="true" gradient="43" />
    <paper height="384" width="165" fid_size="50" />
    <handler min="25" max="150" />
    <mode object="blob" />
    <tiempo segundos="5" />
    <dibujo id="10" />
    <calibration file="default.grid" invert=" " />
    <fiducial engine="amoeba" tree="miniset.trees" />
</reactivision>
```

Figura 41 Ejemplo de fichero de configuración de Reactivision.

Por último se modificó la gestión de los ficheros de configuración por parte de Reactivision para que permita leer también los nuevos parámetros añadidos y a su vez guardase los cambios realizados a través del interfaz de usuario, de manera que no fuera necesario memorizar estos valores para después ser añadidos a mano en los ficheros,

haciendo así mucho más cómodo el manejos al usuario. Para ello se utilizó la clase `TiXmlDocument`, la cual permite un rápido manejo de las etiquetas XML. Las funciones modificadas fueron `readSettings`, función que se encarga de leer y cargar los valores de los parámetros almacenados en el fichero de configuración y `writeSettings` encargada de actualizar y guardar los cambios realizados en los parámetros durante la ejecución.

Capítulo 6. Resultados.

Tras la finalización de cada uno de los objetivos desarrollados durante este proyecto, se han creado juegos para comprobar que cumplían la función de mejora del *framework*.

En este capítulo se presentan los juegos realizados a raíz de las mejoras explicadas en los capítulos anteriores. En el anexo D se hace un estudio y comparación de otros *tabletops* y *frameworks* en el mercado.

6.1 Secuenciador.

El secuenciador es un juego de música en el que los niños pueden crear ritmos de batería usando las clásicas fichas circulares del juego de las damas. La partitura utilizada para crear ritmos musicales, se puede entender como una distribución de notas musicales (las fichas), sobre una superficie bidimensional (la mesa), en el que el eje horizontal representa el tiempo, y el horizontal los diferentes instrumentos de la batería, representados con diferentes colores (ver fig. 42). La partitura se reproduce de izquierda a derecha.

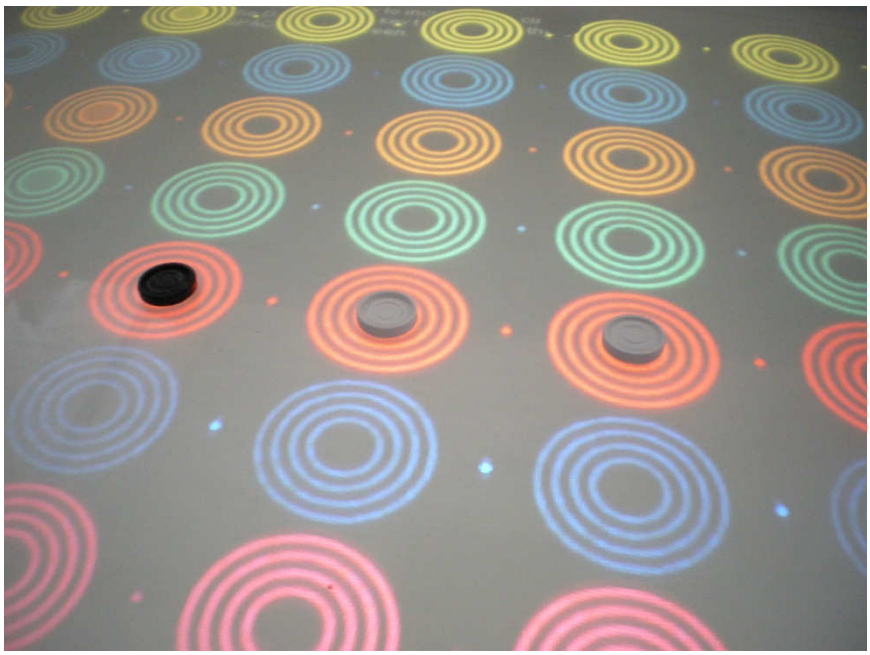


Figura 42 Capturas del juego secuenciador.

Para que las fichas fueran identificadas por Reactivision, se ha usado la nueva colección de fiduciales. Como en el juego de las damas, tenemos dos tipos de fichas, las blancas, y las negras. A estas últimas se les añadió un fiducial de la nueva colección con un único punto negro (ver fig. 43).

Figura 43 Fichas usadas en el juego de música 'Sencuenciador' y los fiduciales asociados a estas

La partitura mostrada en la mesa consiste en una rejilla de 8 instrumentos por 16 tiempos, por lo que sería posible incluir hasta 128 fichas para llenar toda la partitura de sonidos. Esto sería imposible usando fiduciales estándar de Reactivision, ya que dado el tamaño mínimo que deberían tener las fichas, no cabrían todas en la mesa. Gracias a los nuevos fiduciales diseñados, estos pudieron ser añadidos sin problemas a fichas normales de juego de damas, cuyas dimensiones permiten llenar toda la partitura.

Los nuevos fiduciales permiten además que Reactivision distinga entre las fichas blancas y negras. En el juego del Secuenciador, dicha distinción se utiliza para variar el tono del instrumento sobre el que está colocada la ficha. En el caso de los instrumentos de percusión (platos, bombo…) las fichas negras tienen un sonido más agudo y corto que las blancas. En el caso de los instrumentos de cuerda (bajo), se ha utilizado un diseño de fiducial orientable, de forma que girando la ficha sobre la mesa, se varía el tono a lo largo de la escala musical, pudiendo componer una melodía de hasta 16 notas.

Por otra parte el juego permite a grupos de niños crear y compartir sus creaciones, gracias al gran número de fichas de las que disponen, evitando situaciones de dominio de los juguetes por solo un niño. Se trata de un juego muy activo y participativo cuando juegan muchos niños (ver fig. 44)

Figura 44 Niños jugando al juego del Secuenciador.

6.2 Juego de pintar.

En este juego se han utilizado la extracción de características de objetos sin fiducial así como el protocolo de comunicación desarrollado.

El juego de pintar permite que los niños dibujen directamente sobre la superficie del *tabletop* o en folios. Al colocar el folio sobre la superficie de la mesa el dibujo queda proyectado en una esquina de la mesa y los niños mediante unos tampones pueden capturarlos y estamparlos por toda la superficie del *tabletop*.

Como se puede observar en la figura 45, se han empleado pinceles normales y de diferentes tamaños. Gracias a la detección de objetos sin fiducial y al cálculo de su área ha sido posible que cada pincel pinte con un grosor acorde a su tamaño en función de la presión que se ejerza. Sin el consiguiente cálculo del área de los objetos reconocidos habría sido imposible el poder pintar con varios grosores ya que todos los pinceles hubiesen sido reconocidos como el mismo objetos, sin tener en cuenta su tamaño, o por lo menos habría tenido que ser de alguna forma no natural, es decir asociando algún tipo de fiducial a los pinceles que identificasen el tamaño de estos.

Figura 45 Pinceles convencionales de diferente grosos utilizados en el juego de pintar.

En la figura 46 se muestra el escenario y modo de funcionamiento del juego: el juego permite realizar trazos con pinceles de varios grosores e ir variando los colores en los que se pinta.

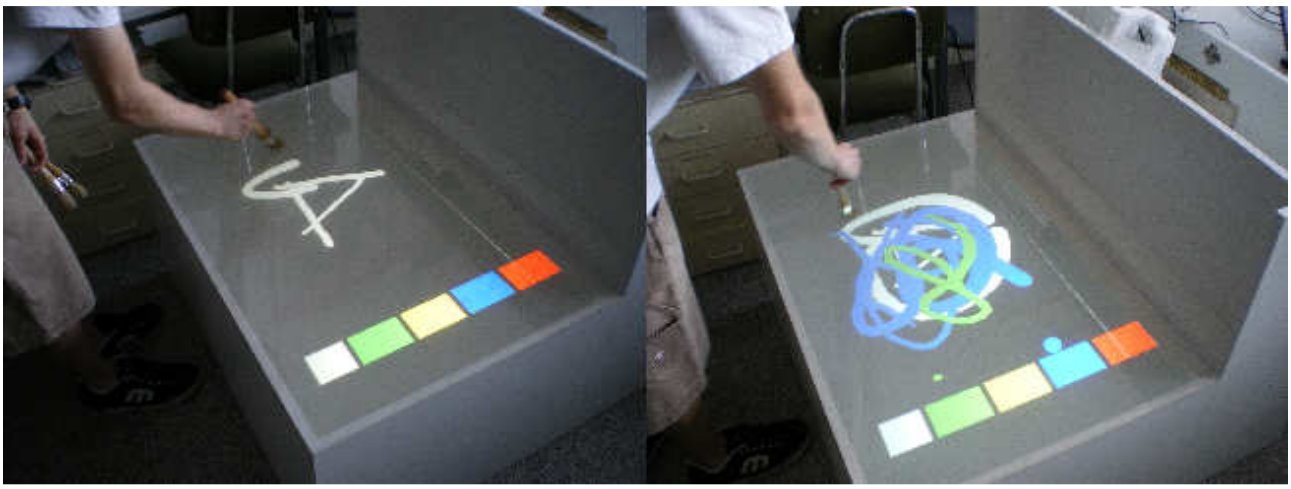


Figura 46 Capturas del juego de pintar

Por otra parte se ha empleado el nuevo protocolo de comunicación y la captura de dibujos para que los niños puedan plasmar en el *tabletop* diferentes dibujos que hayan hecho en papel y así puedan colorearlos y moverlos por la superficie de la mesa (ver fig. 47)

El nuevo protocolo de comunicación hace que el juego sea mucho más fluido eliminando los parones que antes se producían por el bloqueo que Flash generaba en Reactivision. El algoritmo de limpieza implementado se utilizada como filtro sobre los dibujos permitiendo obtener una imagen mucho más nítida y que se parezca más a los dibujos realizados por los niños.

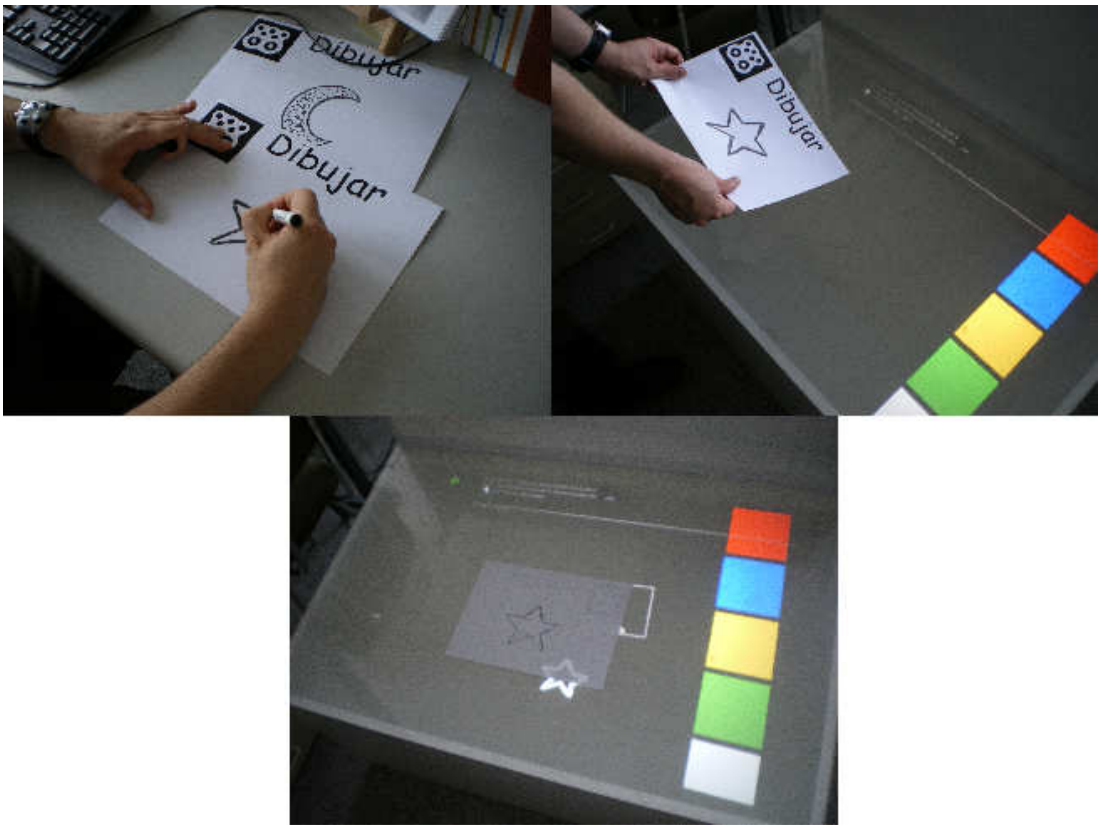


Figura 47 Capturas del juego pintar, en las que se puede ver como son capturados los dibujos en papel realizados por los niños.

6.3 Bugaboo.

Para la realización de este juego se han empleado los algoritmos de extracción de contornos y cálculo de la orientación de objetos sin fiducial, así como el protocolo diseñado para el envío de la información.

El juego consiste en ayudar a una pulga a comer frutas que van apareciendo en el *tabletop* utilizando objetos que una vez reconocidos, se dibujan en el *tabletop* y en los que se puede apoyar la pulga para saltar y así llegar hasta las frutas.

El cálculo de los contornos ha permitido la utilización de cualquier tipo de objeto como plataforma. De otro modo habría sido necesario utilizar juguetes aumentados con fiduciales lo que habría hecho menos versátil el juego, ya que consiste en hacer utilizar la imaginación a los niños para ayudar a la pulga. Pueden utilizar cualquier tipo de objeto: plastilina, maderas incluso la propia mano de los niños.

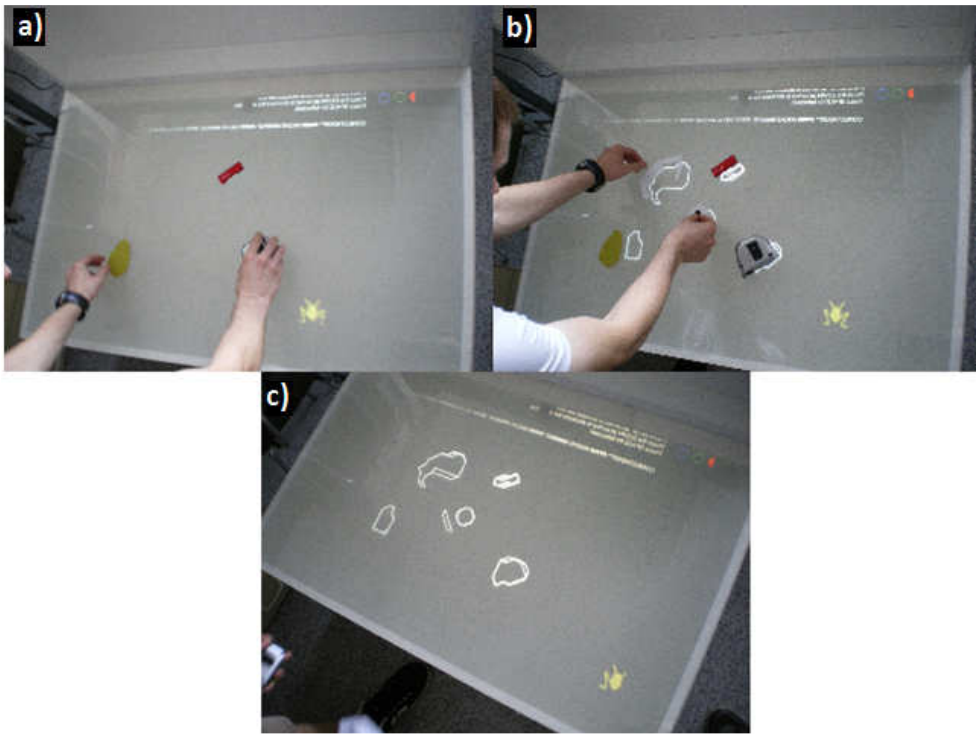


Figura 48 Capturas del juego bugaboo. A) Niño dibujando sobre un folio identificado con un fiducial. B) Colocando los objetos sobre el *tabletop.* C) Contornos de los objetos situados en el *tabletop.*

En la figura 48 se pueden observar imágenes del juego, en las que se muestra como objetos de todo tipo son utilizados a modo de plataforma. En b) y c) se observa cómo es calculado y representado el contorno de estos en el *tabletop*.

Por otra parte se ha utilizado el nuevo protocolo de comunicación implementado, así como la estructura de datos definida para los contornos. De este modo es posible enviar la información de los contornos de los objetos detectados al juego, ya que el protocolo TUIO sólo permite el envío de un limitado número datos con estructuras simples.

6.4 Pruebas reales con niños

Los resultados de este proyecto han sido utilizados y evaluados por niños con edades comprendidas entre los 5 y 10 años. Al finalizar cada una de las funcionalidades diseñadas, familiares de compañeros del departamento era invitados para pasar una tarde jugando con los juegos desarrollados permitiendo así pudiéndose comprobar los puntos fuertes y debilidades que las funcionalidades desarrolladas a lo largo del PFC mostraban.

Además se han podido comprobar los resultados obtenidos en eventos públicos como son “La Noche de los Investigadores” (23 de Septiembre de 2011) y La Semana de la Ingeniería (7 al 11 de Noviembre). Eventos con fines lúdicos donde se da a conocer a todo el mundo los proyectos que se están llevando a cabo en la Universidad.

La Noche de los Investigadores, fue un evento diseñado para que niños de entre 5 y 6 años y sus familias pasaran una tarde de ocio de la mano de algunos proyectos de la universidad. Durante la duración de este evento grupos de cinco y ocho niños iban viniendo a jugar con el *tabletop* NIKvision. Mientras a los padres se les explicaba en qué consistía el funcionamiento de este prototipo, los niños probaban cada uno de los juegos. Este evento nos permitió conocer qué juegos gustaban más y cuales menos, ya que se les dejaba total libertad para cambiar de un juego a otro y expresar sus opiniones.

La Semana de la Ingeniería se trata de un evento que sirve para promocionar la ingeniería entre los más pequeños. Durante la segunda semana de Noviembre niños de diferentes colegios de Zaragoza, con edades comprendidas entre los 8 y 10 años se acercaron a la universidad para ver de primera mano y poder probar los prototipos de las distintas investigaciones que se están realizando en ella. Durante toda la semana grupos de niños tuvieron la oportunidad de jugar con el *tabletop*, descubriendo una forma diferente de jugar con herramientas digitales.

La experiencia obtenida durante estos eventos resultó gratificante y satisfactoria al comprobar el buen funcionamiento de las nuevas funcionalidades (Ver figuras 49, 50 y 51). Además el poder probar las funcionalidades diseñadas en este PFC con niños ha supuesto un feedback que ha permitido mejorar los resultados obtenidos. Por ejemplo en el juego “Bugaboo” se ha reducido los tiempos de envío de información a los juegos, ya que se detectó que estos eran demasiado elevados.

Figura 49 Niños jugando al juego Secuenciador (izquierda) y juego de pintar (derecha). Noche de los investigadores.

Figura 50 Configurando los parámetros de los juegos. Noche de los investigadores.

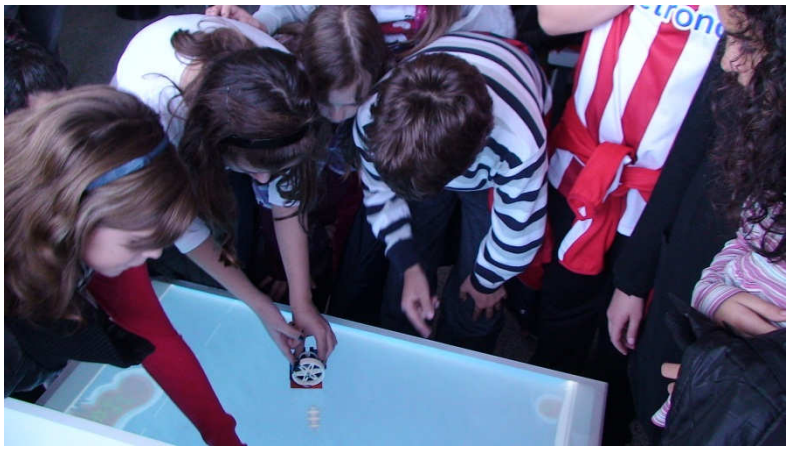


Figura 51 Niños jugando con NIKvision. Semana de la Ingeniería.

Capítulo 7. Conclusiones.

Una vez finalizado el proyecto y obtenidos los resultados se procede a realizar un breve análisis del trabajo realizado en el que se incluye: el grado de cumplimiento de los objetivos principales, valoración y posibles mejoras o trabajo futuro en este ámbito. En el Anexo E se muestra el desarrollo temporal seguido en el PFC.

7.1 Conclusiones

En los requisitos del proyecto se especificaban los objetivos principales que se debían lograr. Como se puede observar a lo largo de la memoria los resultados y los objetivos se han cumplido satisfactoriamente:

1. Se ha diseñado una nueva colección de fiduciales que pueden ser incorporados en juguetes pequeños y con formas no cuadradas. Esto fiduciales permiten distinguir juguetes con la misma forma y tamaño, de manera que puedan utilizarse en juegos en los que cada jugador se identifique con un tipo de ficha.

2. Se han estudiado e implementado diferentes algoritmos de visión para el reconocimiento y seguimiento de los nuevos fiduciales: algoritmos de etiquetado de áreas, eliminación de ruido, detección de agujeros.

3. Se han extraído las características geométricas necesarias para poder identificar y emplear juguetes en los que no se puede incorporar un fiducial, siendo las características contempladas han sido: área, orientación y contorno.

4. Se ha diseñado e implementado un protocolo de comunicación que permite el envío de la nueva información generada en las distintas partes del proyecto: características de los objetos sin fiducial, dibujos en papel, etc, así como el envío de datos con estructuras complejas como el contorno de los juguetes. Además se ha mejorado el envío de los dibujos detectados mejorando las limitaciones debidas a los bloqueos.

5. Se ha modificado la interfaz de usuario de Reactivision para permitir configurar los valores que regulan las nuevas funcionalidades, y se ha mejorado los ficheros de configuración que permitan cargar y almacenar dichos valores.

Además de cumplir los objetivos, todo lo que se ha desarrollado se ha incorporado en diferentes juegos para demostrar su correcto funcionamiento.

7.2 Valoración personal

Desde un punto de vista personal el desarrollo de este PFC ha supuesto:

1. La oportunidad de aplicar los conocimientos adquiridos a lo largo de la carrera y aprender aquellos necesarios para la correcta realización del proyecto.

2. El llevar a cabo un proceso completo de diseño, desde las primeras fases de análisis hasta la posterior implementación de las funcionalidades.

3. El trabajar en un proyecto real, cooperando con un grupo de investigación multidisciplinar en un tema de investigación puntero.

4. La comprobación el resultado obtenido con usuarios reales, lo que ha provocado un feedback del trabajo conseguido. Además el trabajar con niños, un tipo de usuario muy exigente, ha facilitado la mejora de las funcionalidades a través de las conclusiones obtenidas de sus reacciones.

7.3 Trabajo futuro

La realización de este proyecto ha abierto nuevas posibilidades de interacción en NIKVision. En un futuro se podrán crear nuevos videojuegos a partir de las funcionalidades desarrolladas, así como crear otros nuevos que abran a su vez nuevas modalidades de juego.

Las nuevas funcionalidades implementadas durante este proyecto dan pie a nuevas líneas de trabajo ya que actualmente no existen diseños de fiduciales que puedan ser adaptados a la base de otros tipos de juguetes en NIKvision (ver fig. 52). En este caso se debería crear fiduciales con forma de arandela y correspondientes algoritmos para su reconocimiento.

Figura 52 Juguetes que necesitan de nuevos fiduciales, ya que sus bases poseen agujeros que s taparían con los fiduciales actuales

Así mismo sería interesante que las funcionalidades realizadas pudieran ser replicadas en otros populares *frameworks* para *tabletops* como CCV (NUI Group, web), expandiendo así el número de diseñadores para *tabletops* que podrían hacer uso de lo desarrollado en este proyecto.

Por otra parte las nuevas funcionalidades debería dar paso a crear algún tipo de interfaz de usuario mediante el que se puedan generar de forma automática los ficheros de configuración, únicamente a través de la selección sencilla del tipo de juguetes a emplear.

Anexo A. Reactivision y TUIO.

A.1 Estructura de Reactivision

A continuación se analiza la estructura de Reactivision, explicando en qué consiste su funcionamiento y las posibilidades de configuración.

A.1.1 ¿Qué es Reactivision?

Reactivision es un *framework* de visión por computador de código abierto y multiplataforma, pensado para la detección rápida y estable de fiduciales incorporados a objetos físicos así como de acciones táctiles [KB07]. Fue diseñado principalmente como un conjunto de herramientas para el rápido desarrollo de *tabletops* y superficies interactivas. El sistema fue desarrollado por Martin Kaltenbrunner y Ross Bencina en el Music Technology Group de la Universitat Pompeu Fabra en Barcelona, España, como parte del proyecto Reactable, *tabletop* que funciona como un instrumento musical, que se explica más en detalle en el anexo D.

A.1.2 Funcionamiento de Reactivision

Reactivision obtiene la información de una mesa transparente a través de una cámara y la procesa para detectar los objetos marcados con fiduciales o los dedos de las manos, la información extraída se envía después a otras aplicaciones a través del protocolo TUIO, que se explica de apartados siguientes. En la figura 53 se observa un diagrama del funcionamiento de Reactivision: la cámara capta la imagen con los fiduciales situados sobre la mesa, envía los datos a Reactivision que tras procesarlos envía la información extraída mediante TUIO a una aplicación TUI (Tangible User Interface) que la usará, por ejemplo los juegos. Por último se muestran los resultados mediante un proyector en la superficie de la mesa.

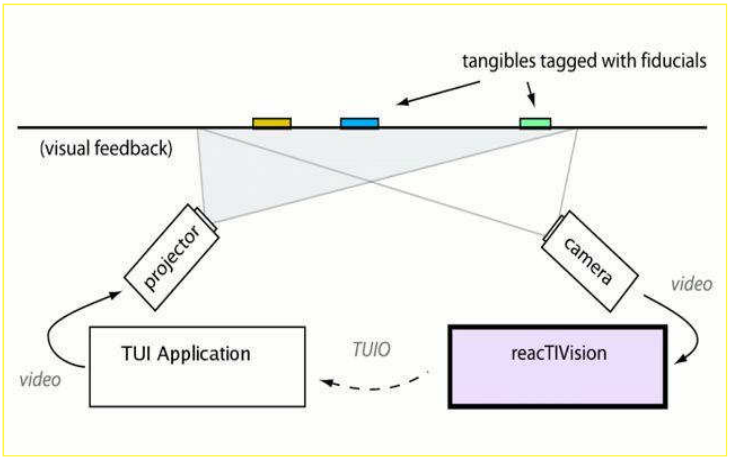


Figura 53 Diagrama general del funcionamiento de Reactivision en un *tabletop.*

Tras capturar la imagen Reactivision la pasa a través de una serie de procesadores que la transforman, para después obtener información de esta. Las fases seis fases que conforman el *framework* en orden de procesamiento son: captura, ecualización, umbralización, detección de fiduciales, calibración y envió de mensajes.

Figura 54 Paso de la información en Reactivision

En la figura 54 se muestra un diagrama del procesamiento de la imagen capturada. Cada una de las fases se encarga de una función en concreto, las cuales se detallan a continuación:

Captura de la imagen: El primer paso consiste en la captura de una imagen y su almacenamiento en un buffer de imágenes.

Ecualización: Es la fase que se encarga de tratar la imagen inicial que le envía la cámara eliminando los elementos erróneos de ésta.

Umbralización: Se encarga de pasar la imagen a escala de grises para convertirla en una imagen binaria, es decir, solo en dos colores, negro y blanco. Para ello realiza una umbralización de la imagen separando el fondo de los objetos, siendo el fondo negro y los objetos de color blanco.

Detección de fiduciales: En esta fase es donde se analiza la imagen para detectar los fiduciales y objetos que se coloquen en la superficie del *tabletop*.

Calibración: Se encarga de corregir la aberración de la cámara que provoca que la geometría de los objetos captados no sea la correcta. Esto se hace a través de una transformación de coordenadas, de manera que exista una correspondencia entre las coordenadas reales y las captadas por la cámara.

Envío de mensajes: Por ultimo toda la información obtenida después del tratamiento se envía a las aplicaciones a través del protocolo TUIO.

En cada ejecución y cuando los valores del fichero de configuración han sido cargados, es cuando se comienza a trabajar en la detección de objetos. En primer lugar se comprueba si la cámara está conectada, de lo contrario se informará de ello mediante un mensaje de error. A continuación se muestra por pantalla la resolución de la cámara y el tipo de esta, así como los diferentes comandos de configuración del *framework*.

En la figura 55 se muestra la ventana de inicio de Reactivision, en la que se observa la fecha de ejecución, el tipo de cámara y la resolución de esta, en este caso 640x480 píxeles. A continuación se pueden ver las diferentes opciones de configuración tanto de los procesadores, como de las variables que el *framework* utiliza.

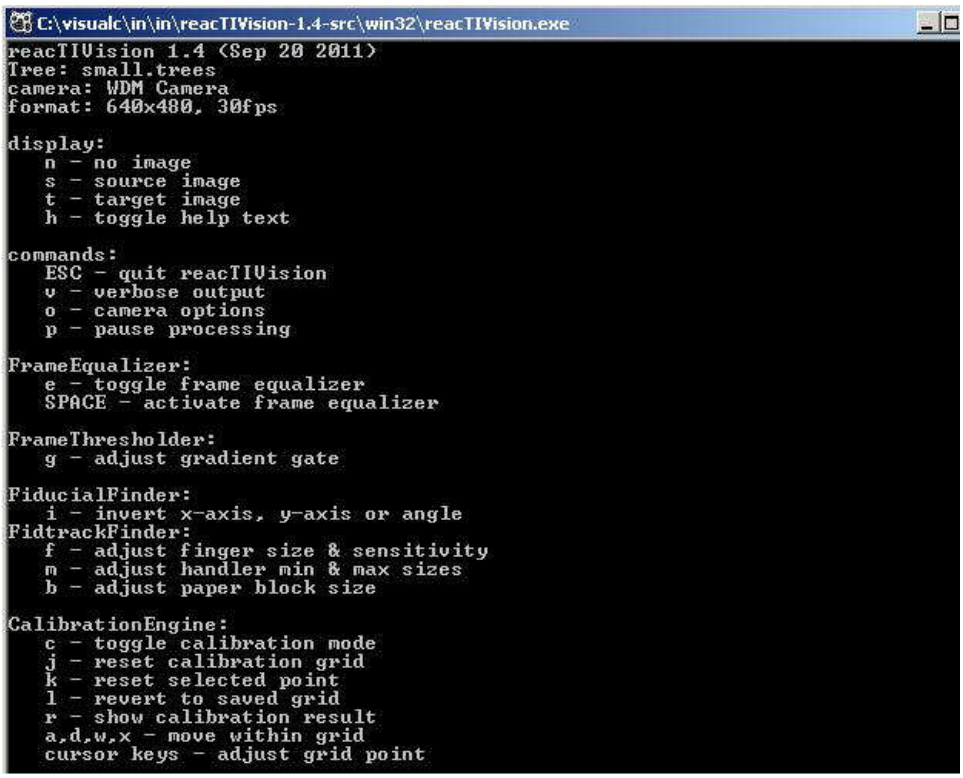


Figura 55 Ventana e inicio de Reactivision

Las fases de ecualización, umbralización, detección de fiduciales y calibración se pueden configurar. Esta configuración, como se ha explicado en el capítulo 5 se guarda después de cada ejecución, de manera que puede volver a recuperarse los parámetros de cada procesador en usos posteriores. La configuración se guarda en un fichero XML, que almacena los valores de las variables de cada procesador, así como el valor de las variables que se vayan a necesitar. Reactivision es el encargado de cargar este fichero de configuración al iniciar su ejecución.

A.1.3 Fases de Reactivision

A continuación se van a explicar más en detalle cada una de las fases que conforman el *framework* Reactivision.

A.1.3.1 Captura de imágenes

La captura de imágenes a través de la cámara se realiza de forma paralela al programa principal, ya que se necesita estar tomando imágenes en todo momento para actualizar los valores de posición, orientación, etc de los objetos así como reconocer los nuevos objetos colocados entre captura y captura. Si la captura de imágenes no se hiciese de forma paralela al procesamiento, el programa principal no podría tatar la imagen en ningún momento o por el contrario al no tomar imágenes continuamente se estarían perdiendo parte de las interacciones con la mesa.

Este proceso captura *frames* y los va guardando en un buffer, pudiendo almacenar 3 imágenes de forma simultánea. Si el buffer se encuentra lleno, se realiza una comprobación del estado de la cámara y si ésta sigue en funcionamiento se espera un tiempo de 5 segundos y después se vuelve a intentar guardar una nueva imagen, de esta manera se evita la saturación del buffer. En caso de que la cámara no responda, se aborta la captura de imágenes.

A.1.3.2 Ecualización

Este es la primera fase en la que se modifica la imagen. Sirve para eliminar objetos erróneos o no necesarios en la imagen. Al capturar por primera vez una imagen, ésta contendrá ruido y errores por la mala iluminación o calibración de la cámara. Para evitar en gran medida la captura de ruido, este proceso indica al *framework* que la primera imagen que tome deberá ser tratada como fondo, de esta manera toda la información que sea errónea será ignorada. Este proceso se puede activar y desactivar pulsando la tecla 'e'. El fondo se puede poner a negro pulsado la barra espaciadora, de esta manera se evitaran falsos positivos en el reconocimiento ya que esta será la imagen tomada como fondo. En la figura 56 se puede ver cómo se elimina el ruido y errores en la imagen pulsando la tecla espaciadora y así indicando al ecualizador que el fondo es todo negro.

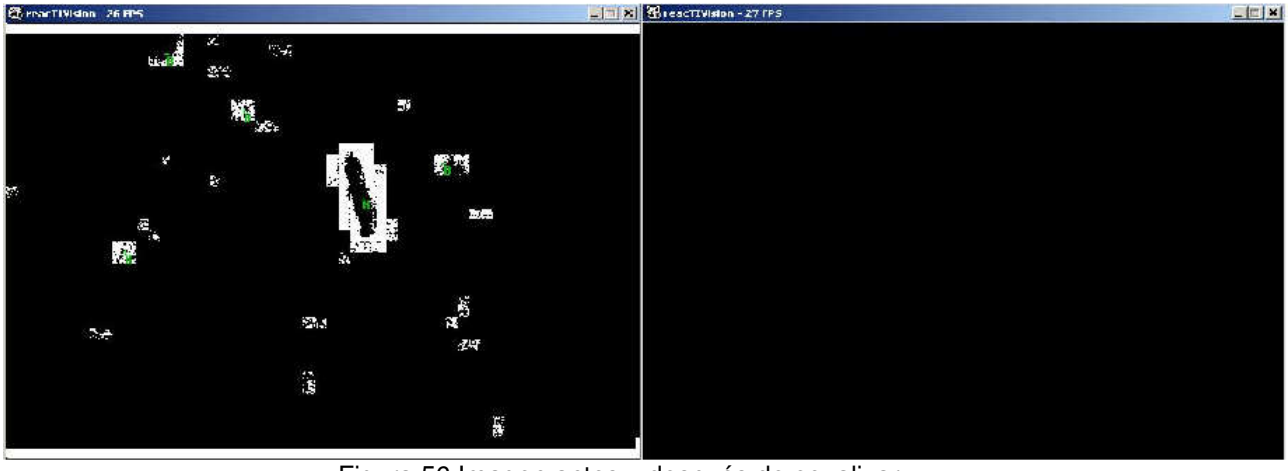


Figura 56 Imagen antes y después de ecualizar.

A.1.3.3 Umbralización

Este fase consiste en umbralizar ecualizada para transformarla en una imagen binaria, es decir en blanco y negro. El procedimiento consiste en ir recorriendo la imagen formando regiones de 6x6 píxeles junto a sus 8 vecinos. En cada región se comprueba el mínimo y el máximo valor de los píxeles y después se compara la diferencia de dichos valores con el valor del gradiente. Si esta diferencia es mayor que el gradiente, se considera que la región pertenece al fondo y por tanto se le asigna el color negro. Si por el contrario es menor que el gradiente, se comprueba si esta diferencia es menor que el valor 127, valor medio del total de escala de grises, si es menor pasa a ser blanco y sino a negro. El valor del gradiente puede modificarse pulsando la tecla 'g'. En la figura 57 se muestra una imagen ecualizada y el resultado tras ser umbralizada.

Figura 57 Imagen antes y después de umbralizar.

A.1.3.4 Detección de fiduciales

Tras la umbralización de la imagen, se pasa a buscar los fiduciales y los objetos sin marcar. Para la identificación, lo primero que se realiza es una segmentación de la imagen, es decir una división de la imagen en regiones. Se comienza recorriendo la primera línea de la imagen pixel a pixel. En el momento que se detecta un pixel, se crea una nueva región si éste es de distinto color al anterior pixel o actualizamos la región del pixel anterior si es del mismo color. Tras analizar la primera línea, se pasa a realizar el mismo proceso con las siguientes líneas, con la salvedad de que ahora también se comprueba el pixel superior al que se está analizando, además del anterior. Cuando un pixel comparte región tanto con su pixel superior como el de su izquierda, se procede a una fusión de sus regiones, dando lugar a una nueva [BKM05].

Una vez segmentada la imagen, se procede a la búsqueda de los fiduciales que coincida con los almacenados en el fichero de árboles que contiene el *framework* al ejecutarse. Una vez reconocidos todos los fiduciales colocados sobre la superficie de la mesa, se comprueba que estos fiduciales sean válidos, nuevos o si ya habían sido reconocidos. Para ello lo que el algoritmo hace es comprobar los fiduciales reconocidos con los reconocidos en el *frame* anterior. Uno a uno comprueba los fiduciales con los ya reconocidos, si se produce una coincidencia se compara la distancia entre los dos y si es menor que el tamaño mínimo de un fiducial, se entiende que se trata del mismo objeto y que simplemente ha sido movido de una

frame a otro, por lo que se procede a actualizar sus valores de posición y orientación. Si por el contrario no coincide con ninguno de los fiduciales ya detectados, se añade este a la lista de nuevos fiduciales.

Una vez detectados los fiduciales convencionales de Reactivision, se pasa a buscar los desarrollados en este proyecto. Para ello se busca en las imágenes segmentadas, regiones del tamaño establecido por parte del usuario, para después realizar un etiquetado de las zonas según su color como se ha explicado en el capítulo 2. Tas el etiquetado se realiza una detección de agujeros en estas zonas. Y por último para cada nuevo fiducial, se comprueban si sus agujeros cumplen las restricciones de tamaño, así como si cumple las restricciones de área blanca que debe tener.

Para evitar falsos positivos y no confundir los nuevos fiduciales con los ya detectados (tanto de la nueva colección como de los convencionales), se procede a realizar una comprobación similar a la explicada en el párrafo anterior.

Por último, se buscan los objetos sin fiducial y posibles *fingers*. Para ello se examinan las regiones de color blanco y de un determinado. Los tamaños que pueden tener tanto los objetos como los *fingers* se puede modificar a través del interfaz de usuario como se ha indicado en el capítulo 5. Una vez detectada las regiones del tamaño establecido, se comprueba que no se trata de objetos que ya se habían detectado anteriormente, para ello se compara la distancia entre cada uno de los nuevos objetos y los ya detectados. Por último y una vez detectados todos los objetos sin fiducial, se procede a extraer las características de estos (área, orientación y contorno) tal y como se ha explicado en el capítulo 3.

Tras haber reconocido todos los objetos situados en la superficie de la mesa, se procede a identificarlos gráficamente en la pantalla de Reactivision. Los fiduciales serán identificados con un número, según su posición en el fichero de árboles antes mencionado. Los dedos de las manos se mostrarán con la letra 'F' y los objetos sin fiducial con la letra 'B'. En la figura 58 se muestra un ejemplo de los objetos que se han reconocido identificados con las correspondientes letras y números.

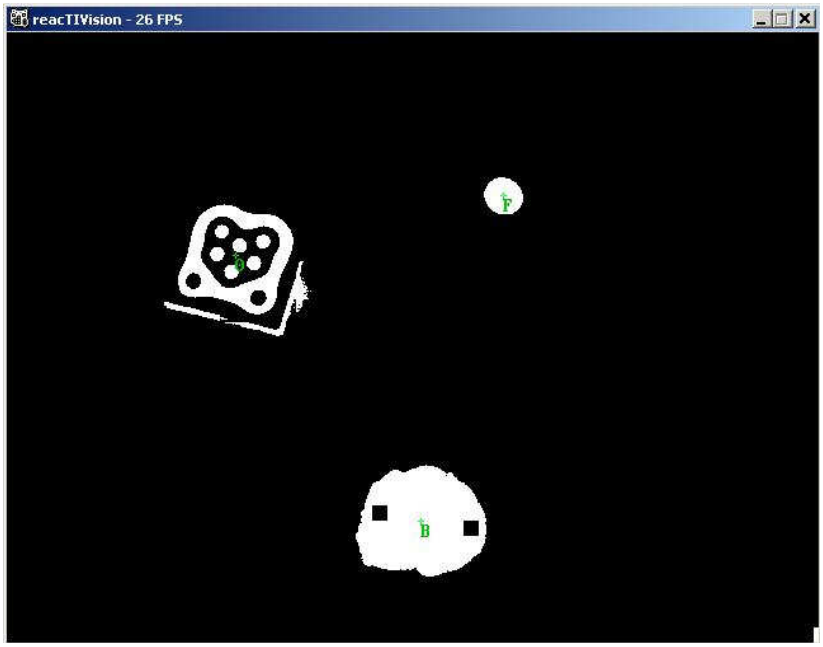


Figura 58 Objetos reconocidos en Reactivision

A.1.3.5 Calibración

Por último se procede a la calibración de la imagen. Esta fase consiste en una conversión entre las coordenadas reales y las obtenidas por la cámara estableciendo así una relación entre ellas. Esto es necesario ya que las cámaras producen aberraciones ópticas, distorsionando la imagen respecto a la realidad, por lo que a la hora de representar elementos en unas coordenadas determinadas, este objeto puede no quedar situado en la posición esperada. El calibrador puede activarse pulsando la tecla 'c'. Al hacerlo se muestra una rejilla formada por líneas y puntos que se pueden mover por la pantalla. El desplazamiento y modificación de los puntos provoca que las zonas de la imagen dada por la cámara que coincidan con esos puntos, se deformen hasta los nuevos puntos establecidos. En las figuras 59 y 60 se observa una imagen sin calibrar y el efecto que se produce al mover los puntos de la rejilla para ajustarlos a la imagen real.

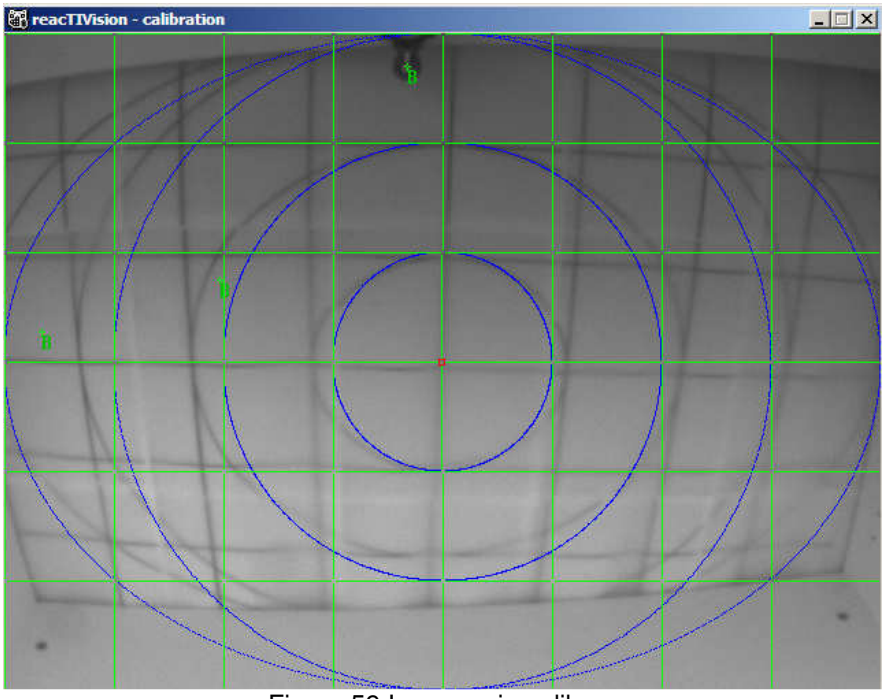


Figura 59 Imagen sin calibrar

Las líneas azules y verdes corresponden al calibrador. Las líneas en negro pertenecen a la imagen real y de cómo tendrían que ser las líneas del calibrador. Se puede ver que no coinciden unas con otras.

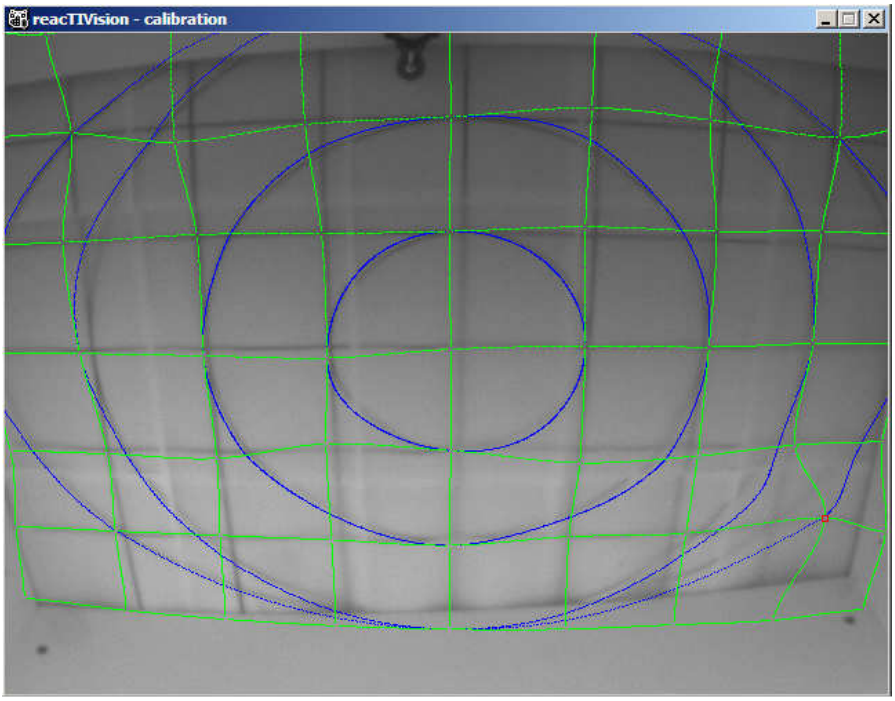


Figura 60 Imagen calibrada

Después de calibrar, se puede comprobar que las líneas negras corresponden con las del calibrador.

A.1.3.6 Envío de mensajes

Por último toda la información obtenida después del tratamiento se envía a las aplicaciones a través del protocolo XML explicado en el capítulo 4 y el protocolo TUIO detallado en los apartados siguientes.

A.2 Estructura TUIO

A continuación se analiza la estructura del protocolo de comunicación TUIO, explicando en qué consiste su funcionamiento y la información que puede enviar.

A.2.1 ¿Qué es TUIO?

Se trata de un protocolo diseñado específicamente para cumplir las necesidades de comunicación que presentan los *tabletops*. Este protocolo define las propiedades comunes que presentan los objetos identificados con fiduciales, así como los movimientos del dedo realizados por el usuario. Su función principal es el envío de la información extraída por las aplicaciones de reconocimiento a las aplicaciones que vayan a usar esta información, en el caso de este proyecto los juegos. TUIO se basa en Open Sound Control- OSC, un estándar emergente para entornos interactivos, diseñado en un principio para *tabletops* musicales. Inicialmente TUIO fue diseñado junto con Reactivision para el proyecto Reactable [Reactable, web], *tabletop* diseñado en la Universidad Pompeu Fabra, que se explica en el anexo D. Hoy

en día sin embargo este protocolo es utilizado por otros proyectos como CCV, *framework* para *tabletops* diseñado por el NUI Group [NUI, web], o la librería touchlib.

A.2.2 Como funciona TUIO

TUIO se basa en la comunicación cliente-servidor. El servidor, en este caso la aplicación de reconocimiento, envía mensajes con la información extraída al cliente, en nuestro caso los juegos en Flash. Cada vez que Reactivision procesa un *frame* y obtiene la información de los objetos situados en la superficie del *tabletop*, envía un mensaje TUIO al juego ejecutado en ese momento.

Este protocolo define dos clases principales de mensajes: mensajes *Set* y mensajes *Alive*. Los mensajes *Set* son usados para comunicar la información de los objetos reconocidos, como la posición, orientación y otras posibles características. Por su parte los mensajes *Alive* indican el estado actual de los objetos situados en el *tabletop*, usando una lista compuesta de identificadores únicos. Para evitar posibles errores producidos por la pérdida de paquetes, no son utilizados mensajes que indiquen explícitamente la adición o el borrado de objetos en la lista. El cliente debe calcular la vida de los objetos comparando la diferencia entre los mensajes consecutivos *Alive* [KBBC05]. Existe un tercer tipo de mensaje denominado *Fseq*, que sirve para agrupar la información que se va a enviar y que contiene un identificador único correspondiente al *frame* que se ha analizado en ese momento€.

TUIO se basa en mensajes UDP. Al no ser un tipo de comunicación fiable, aumenta la posibilidad de la pérdida de paquetes. Para evitar esto, en cada envío, se incluye la información de todos los objetos detectados, incluso si no han cambiado respecto del *frame* anterior. De esta manera el protocolo asegura que no se pierda información aunque se pierda algún paquete.

A.2.3 Datos del paquete TUIO

En cada envío se incluye las características de cada objeto detectado. Los datos añadidos son los siguientes:

1. Identificador de sesión: Identificador único que identifica a cada objeto y es asignado durante el reconocimiento de estos.

2. Identificador de clase: Número del fiducial del que se envía la información. En el caso de ser un objeto sin fiducial se envía el identificador 500, para indicar al juego de que se trata de un objeto de este tipo.

3. Posición: Coordenadas (x,y) de pantalla del objeto detectado. Son coordenadas con valores comprendidos entre 0 y 1, así se asegura la normalización de estos valores para cualquier tipo de resolución.

4. Orientación: Ángulo de orientación del objeto reconocido.

5. Tamaño: Alto y ancho del objeto detectado.

6. Vector de velocidad: Entero que representa la velocidad de desplazamiento en x e y de un objeto basándose en la posición anterior y en la actual.

7. Vector de orientación: Número en coma flotante que representa la velocidad de rotación de un objeto basándose en la orientación anterior y en la actual.

8. Parámetro libre: Espacio libre en el paquete para poder enviar información adicional.

Tanto el vector de velocidad como el de orientación, son datos nulos, ya que aunque hay espacio reservado para ellos, no se calculan.

A.2.4 Formato de los paquetes TUIO

Para que la información pueda ser tratada con facilidad, todos los paquetes TUIO, sean del tipo que sean, tienen un formato específico. De esta manera la aplicación cliente puede decodificar estos paquetes sin problema. En la figura 61 se observa ver un esquema de la estructura que siguen estos paquetes.

Figura 61 Estructura de los paquetes TUIO

La cabecera del paquete especifica si la información enviada corresponde con un objeto identificado con un fiducial o de otro tipo. El segundo dato indica el tipo de mensaje del que se trata: *Set*, *Alive* o *Fseq*. Dependiendo del tipo de mensaje el campo datos contendrá una información u otra:

1. Mensaje SET: Identificador de la sesión y parámetros del objeto identificado.

2. Mensaje ALIVE: Lista que contiene los identificadores de sesión de todos los objetos que se encuentran situados en la superficie de la mesa.

3. Mensaje FSEQ: Identificador del *frame* tratado.

Anexo B. Análisis

En este anexo se explica la metodología de análisis, en la que se definen los requisitos y los diagramas utilizados, así los pasos seguidos para desarrollar cada modelo de descripción del sistema.

B.1 Metodología de análisis

La metodología de análisis seleccionada es OMT [RBPEL98]. Según esta metodología la fase de análisis se realiza en tres pasos:

1. Modelo de objetos: Describe la estructura estática de los objetos del sistema (relaciones, atributos y operaciones), el cual se representa mediante diagramas de clases.

2. Modelo dinámico: Describe los aspectos de un sistema estudiando la organización de estados y la secuencia de operaciones asociadas con los usuarios.

3. Modelo funcional: Describe las transformaciones que pueden sufrir los datos dentro del sistema, representado gráficamente mediante diagramas de flujo de datos.

Aunque el análisis de requisitos es propio de la metodología UML, se ha creído oportuno añadirlo a estas tres etapas de la fase de análisis, ya que permitirá estudiar más a fondo las necesidades del *framework*. Por otra parte el modelo dinámico se ha omitido por ser prácticamente nula la interacción directa con el usuario y las partes implementadas en este proyecto.

B.2 Análisis de requisitos

En esta sección se muestra la lista de requisitos funcionales (RF) y no funcionales (RNF) que ha de cumplir el sistema y que se pueden ver en la siguiente tabla.

REQUISITOS FUNCIONALES

RF-1 Se podrán identificar en Reactivision, juguetes o fichas de juego de dimensiones menores de 4x4 cm.

RF-1.1 Se podrá identificar hasta 4 tipos de juguetes o fichas distintos.

RF-1.2 Se extraerá información de la orientación de los juguetes situados en la mesa.

RF-2 Se podrán identificar objetos planos (sin fiducial), situados en la superficie de la mesa.

RF-2.1 Se extraerá el área de los objetos planos situados en la mesa.

RF-2.2 Se extraerá la orientación de los objetos planos situados en la mesa.

RF-2.3 Se extraerá el contorno de los objetos planos situados en la mesa.

RF-3 Se creará un protocolo de comunicación, que permita mandar la información que se obtiene de RF-1 y RF-2, pero que permita fácilmente ampliarse en el futuro con otras informaciones.

RF-3.1Se podrá enviar a los juegos información con estructuras complejas, no limitándose al envío de datos simples como: enteros, flotantes, caracteres , etc.

RF-4: Permitir al usuario de Reactivision parametrizar las nuevas funcionalidades creadas en RF1 y RF2 a través de la interfaz gráfica de Reactivision.

RF-4.1 Se deberán poder inicializar los nuevos parámetros a través de los ficheros de configuración de Reactivision.

RF-4.2 Se guardaran los cambios realizados durante la ejecución del juego en los ficheros de configuración de Reactivision.

REQUISITOS NO FUNCIONALES

RNF-1 Las nuevas funcionalidades deberán ser robustas frente a distintas condiciones de iluminación y ruido de cámara.

RNF-2 Dado que las nuevas implementaciones de reconocimiento visual y protocolos de comunicación van a ser usadas en juegos en los que el usuario interacciona en tiempo real, los tiempos de procesado deberán mantenerse bajos para que la respuesta del juego al usuario sea fluida.

RNF-3. La herramienta de trabajo será visual C++.

B.3 Modelo de objetos

Como se ha explicado anteriormente el modelo de objetos define la estructura estática de los objetos del sistema y proporciona el entorno en el que situar el modelo funcional.

En esta sección se muestra el diagrama de clases que componen la aplicación así como el glosario de términos. Primero se hace un estudio de las principales clases mostrando las relaciones que existen entre todas ellas, y después se explica en detalle los atributos de éstas y sus funciones.

Se ha añadido también un glosario de los términos más usados en esta memoria.

B.3.1 Diagrama de clases

Una vez realizado el análisis de las clases de las que se compone el sistema, se representa esquemáticamente cada clase y se muestran las relaciones existentes entre ellas.

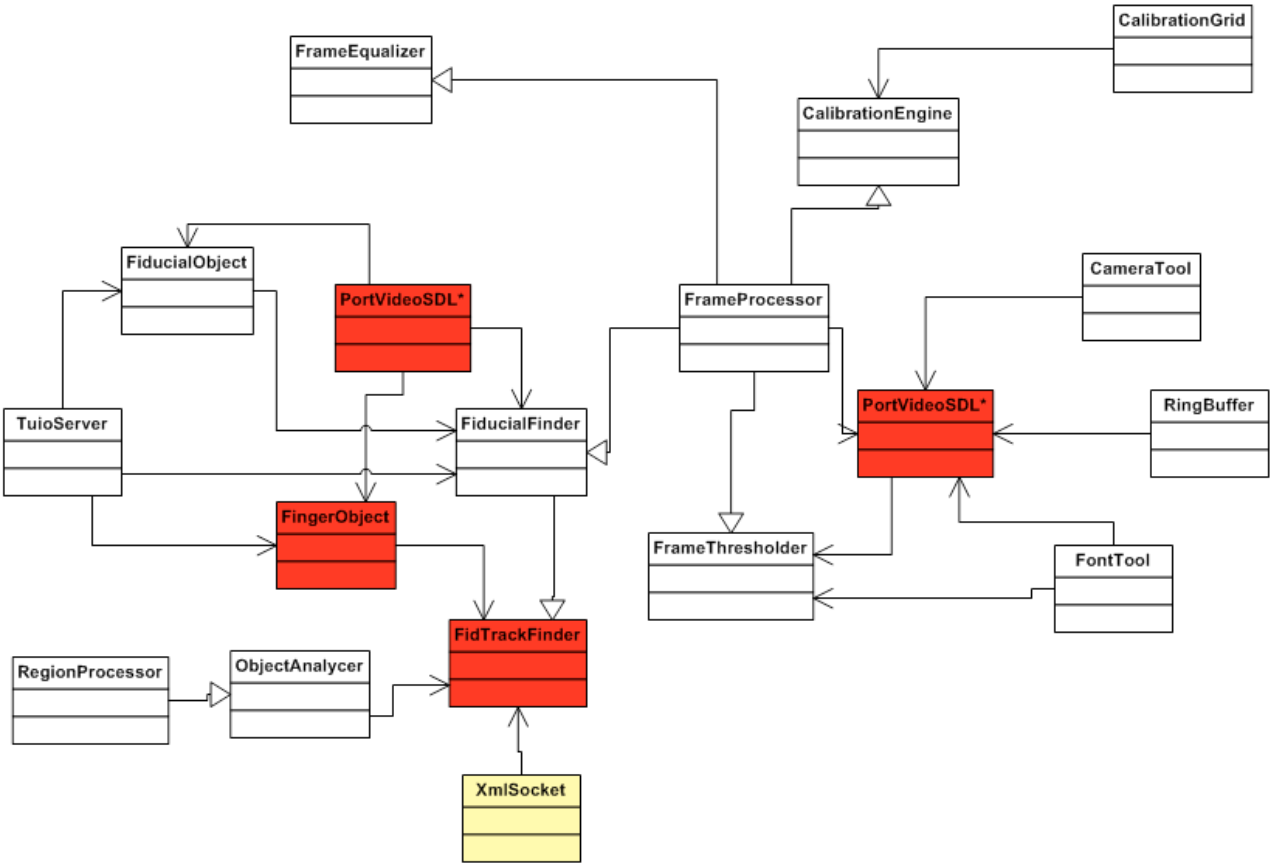


Figura 62 Diagrama de clases de Reactivision

En la figura 62 se muestra la relación entre las clases de Reactivision, siendo el origen de la flecha la clase que proporciona información a la clase a la que le llega la punta de la flecha. Las flechas con punta huecas indican herencia, la clase a la que la flecha llega hereda de la flecha de la que sale la flecha. Por otro lado las clases que tienen el fondo de color rojo son las que se han modificado, mientras que las coloreadas en amarillo son las creados por completo en el desarrollo del proyecto.

En el apartado siguiente se especificarán las clases modificadas o creadas a lo largo de este PFC.

B.3.2 Funciones y atributos

A continuación se explican los atributos y funciones de las clases modificadas o creadas en este proyecto. Las funciones y atributos recuadrados, corresponden a las partes modificadas o creadas a lo largo de este proyecto.

PortVideoSDL

Clase que actúa como el motor de tratamiento de la información de todo el *framework*.

Atributos:

Camera_: Dirección de memoria de la cámara.
CameraBuffer_: Buffer donde se almacena lo leído por la cámara.
Camera_config: Nombre del fichero de configuración de la cámara.
Running_, error_, pause_, calibrate_, help_: Booleanos que indican si el *framework* se encuentra ejecutándose, si ha habido un error, si está pausado, calibrando o mostrando la ayuda.
Framenumber_: Número de frame se que está tratando.
RingBuffer: Buffer que almacena hasta 3 *frames* indicando cual es el siguiente que debe ser leído.
Current_fps: *Frames* por segundo a los que está trabajando la cámara
Display_lock: Booleano que indica si el acceso al *display* está siendo usado por algún procesador o si se encuentra libre.

dibujOk: booleano que sirve para indicar que el dibujo ha sido reconocido y pude ser tratado
imagen_limpia: Imagen que contiene el dibujo después de ser tratado.
anchura, altura: Valores que indican la altura y anchura del dibujo.

Métodos:

PortVideoSDL(const char* name, bool background, const char* camera_config): Constructor de la clase.
Run(): Activa el funcionamiento del *framework* Reactivision
Stop(): Desactiva del funcionamiento del *framework*
AddFrameProcessor(FrameProcessor *fp): Añade un procesador de *frames* al motor.
***MainLoop*():** Ejecuta el bucle principal en el que se realizan todas las acciones de tratamiento de la información de las imágenes. En este método se ha añadido el proceso de tratamiento del dibujo: rectificación de la imagen, recorte y filtro de media.
RemoveFrameProcessor(FrameProcessor *fp): Eliminar un procesador de *frames* del motor.
SetMessage(std::string message): Escribe el mensaje por la consola de comandos.
DisplayMessage(const char *message): Escribe el mensaje en el *display* del *framework*.
SetDisplayMode(DisplayMode mode): Establece que está mostrando el *display*: la imagen original, la imagen umbralizada o nada.
GetDisplayMode(): Devuelve el modo en el que se está mostrando el *display*.
CurrentTime(): Devuelve el tiempo que lleva funcionando el *framework*.
SetupWindow(): Inicializa la ventana del *display* en la que se mostrará la información de la cámara y de la detección.
TeardownWindow(): Elimina la ventana del *display*.

SetupCamera(): Inicializa la cámara para que proceda a la captura de imágenes.
TeardownCamera(): Deja de usar la cámara.
InitFrameProcessors(): Inicializa todos los procesadores de *frames* que tiene añadidos. Si alguno no se puede inicializar, lo elimina.
AllocateBuffers(): Establece el tamaño de los buffers de origen, destino y *display* y los reserva en memoria.
FreeBuffers(): Elimina de memoria el espacio reservado para los buffers.
EndLoop(): Finaliza la ejecución de los procesadores de información, mostrando por pantalla un mensaje de error si han finalizado por esa razón.
Process_events(): Gestiona el manejo de eventos, que en este caso son las acciones que activan la pulsación de las teclas de opciones.

FidtrackFinder

Clase que se encarga de buscar fiduciales y objetos sin identificador en los *frames*.

Atributos:

detect_finger: Booleano que active o desactiva la detección de dedos.
average_leaf_size: Indica el tamaño de los nodos hoja de los fiduciales.
average_fiducial_size: Indica el tamaño que deben tener los fiduciales.
average_finger_size: Indica el tamaño que deben tener los dedos de las manos para ser reconocidos.
finger_sensitivity: Indica la tolerancia de error del tamaño de los dedos para ser reconocidos.
num_blancos: Número de píxeles blancos que debe tener un fiducial de la nueva colección.
min_handler_size: Tamaño mínimo que debe tener un objeto sin identificador con un fiducial.
max_handler_size: Tamaño máximo que debe tener un objeto sin identificador con un fiducial.
paper_height: Indica la altura del folio que contenga el dibujo a reconocer.
paper_width: Indica la anchura del folio que contenga el dibujo a reconocer.
fid_size: Tamaño que debe tener el fiducial que identifica al dibujo a reconocer.
Modo: Indica el modo de reconocimiento: *blob* si debe reconocer objetos sin fiducial y finger2 si tiene que reconocer fiduciales de la nueva colección.
tiempo_xml: Indica el número de segundos que deben pasar entre envíos del fichero XML que contiene los datos extraídos.
fid_dibujo: Identificador del fiducial que acompaña al dibujo.
Socket: Socket que conecta a Reactivision con el juego.
anterior, ahora: Variable que nos indican el tiempo que ha pasado desde el último envió del fichero XML.
setFingerSize, setFingerSensitivity, setNumBlancos, setHandlerSize, setMinimun, setPaperBlockSize, setHeight, setWidth, setFiducialSize: Booleanos que indican que valor a de aparecer en el *display* para su configuración.
Detectado: Booleano que indica que fiducial que acompaña al el dibujo ha sido detectado.

Tiempo: Tiempo que debe pasar el dibujo sin moverse para ser capturado.

Guardado: Indica que el dibujo ha sido reconocido y guardado.

Métodos:

FidtrackFinder(MessageServer *server, const char* tree_cfg, const char* grid_cfg, int finger_size, int finger_sens): Constructor de la clase que da valor al servidor que envía los mensajes con lo detectado (server), al fichero de configuración de los árboles de los fiduciales (tree_cfg), a la superficie calibrada (grid_cfg), al tamaño de los dedos que se van a detectar (finger_size) y a la variación de tamaño que se establece para detectar un dedo (finger_sens). Además realiza la llamada a la conexión socket, en el caso de que esta opción este activada.

Init(int w ,int h, int sb, int db): Inicializa el procesador de detección.

GetFingerSize(): Devuelve el valor del tamaño de los dedos que se quieren detectar.

GetFingerSensitivity(): Devuelve la variación de tamaño que puede tener un dedo para que, aunque no sea del tamaño exacto, se pueda detectar.

Process(unsigned char *src, unsigned char *dest, SDL_Surface *display): Realiza la segmentación y procesado de la imagen umbralizada para detectar fiduciales y dedos y objetos sin fiducal y muestra los datos por el *display*. En este método se han realizado modificaciones para realizar la detección del fiducial de la captura de dibujo, contabilizar el tiempo que está quieto y enviar la información para que la use la aplicación juego. También se ha realizado el cálculo de la orientación de los objetos sin fiducia, así como el cálculo del área y contorno de estos. Una vez calculado todo se procede al envío de la información a través del protocolo diseñado usando sockets y el fichero XML, todo ello explicado en el capítulo 4.

DrawGUI(SDL_Surface *display): Dibuja en el *display* la información sobre el tamaño de los dedos que se van a detectar para que se pueda modificar, mostrando el tamaño que deben tener para ser reconocidos. Aquí se han realizado modificaciones para poder mostrar las opciones de configuración del tamaño de la zona de captura del dibujo (con sus máximos y mínimos) así como otras variable para el reconocimiento de la nueva colección de fiduciales como el número de píxeles blancos que deben tener. Las opciones son mostradas como una barra horizontal en la que se aumenta o disminuye su valor y el tamaño del dibujo a través de una grafica.

ToggleFlag(int flag): Realiza la gestión de teclas para la configuración del detector. En la gestión se han incluido también las teclas para lanzar la configuración (m y b). Cuando se están modificando los valores de configuración no se puede llamar a ninguna de las otras opciones de modificación de Reactivision hasta que no se termine de darles valores.

limpia_imagen(int **imagen_sucia, int **imagen_limpia, int ancho, int alto): Realiza un filtro de media a imagen_sucia y el resultado es almacenado en imagen_limpia. Sirve para eliminar el ruido de la imagen tomada y poder enviar un resultado optimo en el fichero XML.

siguientMovimiento(int actual_i, int actual_j, int anterior_i, int anterior_j): Dadas las coordenadas del pixel actual y el anterior, devuelve el siguiente movimiento realizar en la búsqueda de los píxeles del contorno.

siguienteVertice(int **vertices, int i, int j, int numVertices): Devuelve el siguiente vertice al de coordenadas (i,j) en sentido horario.
ordenar_píxeles(int **vertices, int numVertices, int **verticesOrdenados): Ordena los vértices en sentido horario y almacena el resultado en verticesOrdenados
busca_nodos(int **verticesOrdenados, int numVertices, int ancho, int alto, int **verticesContorno): Busca los nodos que formaran los segmentos del contorno, a partir de los píxeles del borde del objeto.
sonParalelos(int x1, int y1, int x2, int y2, int x3, int y3): Devuelve cierto si los segmento (x1,y1 – x2,y2) y (x2,y2 – x3,y3) son paralelos. En caso contrario devuelve falso.
etiquetar_matriz(int **matriz_limpia, int **matriz_etiquetada, int ancho, int alto): Realiza el etiquetado de las áreas negras de matriz_limpia y el resultado lo devuleve en matriz_etiquetada

FingerObject:

Clase que representa un objeto sin fiducial o un dedo de la mano con sus propiedades. Con la nueva implementación realizad además también representa a los *blobs* mano y objetos planos.

Atributos:

Alive: Indica si el objeto reconocido se encuentra activo en el último frame analizado.
Unsent: Indica si el objeto reconocido está siendo detectado, pero todavía no se ha enviado información sobre él.
Session_id: Identificador de sesión del objeto reconocido.
State: Estado del objeto reconocido: añadido, borrado, expirado o vivo.
Smallest_area: mínima área que el objeto reconocido tiene que tener para ser detectado.
Xpos, ypos: posición en pantalla del objeto reconocido. Además se ha incluido un campo que contiene la información sobre la orientación.

Métodos:

FingerObject(int width, int height): Constructor de la clase usando como parámetros su altura y anchura (width, height).
Update(float xpos, float ypos, float area): Actualiza la información del objeto reconocido en pantalla: su nueva posición y su nueva área.
update(float xpos, float ypos, float area, double orientacion): Actualiza la información del objeto reconocido en pantalla: su nueva posición,su nueva área y su orientación.
RedundantSetMessage(TuioServer *server): Crea un mensaje para enviarlo mediante protocolo TUIO cuando no se ha modificado ninguno de sus parámetros.
GetStatistics(): Devuelve una cadena con toda la información del objeto reconocido.
CheckStatus(int s_id): Devuelve el estado del objeto reconocido indicado (s_id). Este estado puede ser: añadido, borrado, expirado o vivo.
Update(float xpos, float ypos, float area, float orientacion): Actualiza la información del objeto reconocido, en la que también se incluye la orientación.

AddSetMessage(TuioServer *tserver): Crea un mensaje para enviarlo mediante el protocolo TUIO a otra aplicación cada vez que el objeto reconocido se actualiza. En el mensaje también se añade la información sobre la orientación del objeto reconocido.
Distance(float x, float y): Calcula la distancia entre un punto de la pantalla (x, y) y el centro del objeto reconocido.
Reset(): Inicializa la información del objeto reconocido.
GetX(): Devuelve la posición horizontal del centro del objeto reconocido.
GetY(): Devuelve la posición vertical del centro del objeto reconocido.
GetOrientacion(): Devuelve la posición orientación de objeto reconocido, sólo para objetos sin fiducial.

XmlSocket:

Clase que representa al socket que conecta a Reactivision con los juegos en Flash. Sirve para el envío del fichero XML que contiene los datos extraidos de los objetos reconocidos.

Atributos

sendBuff: Buffer que almacena la información a enviar. **fd:** Identificador del socket.

Métodos:

XmlSocket(): Constructor de la clase. **Run():** Envía la política de conexión al juego y le indica el puerto del socket al que se tiene que conectar (3335). **Conexion():** Crea el socket y se queda esperando a que el juego se conecte. **Envia(char sendBuff[]):** Envía el fichero XML generado al juego.

B.3.3 Glosario de términos

El diccionario de datos contiene una lista de conceptos básicos necesarios para la correcta interpretación de algunos de los aspectos del proyecto.

Blob: Elemento de una imagen que viene determinado por ser un conjunto de píxeles de un color en una región discreta, rodeada por píxeles de otros colores. En el caso de este proyecto los *blobs* son binarios, o blancos, o negros.

Fiducial: Imagen usada para identificar un objeto mediante un sistema de detección visual. Esta imagen puede dar información, sobre el tipo de objeto que es, la orientación y la posición del objeto. La imagen consiste en un conjunto de elementos blancos y negros.

Frame: Imagen particular que se encuentra dentro de una sucesión de imágenes que componen una animación o video. La sucesión continua de estas imágenes producen sensación de movimiento, provocado por las pequeñas diferencias que hay entre cada una de estas.

Framework: Estructura software de soporte en la que otro proyecto de software puede ser organizado y desarrollado. Esta estructura permite facilitar el desarrollo del software, evitando los detalles de bajo nivel para poder destinar tiempo y esfuerzo a identificar los requerimientos que tendrá lo desarrollado.

Momento: Parámetros de una imagen que están relacionados con el tamaño, la posición, la orientación y la forma de ésta. Los momentos de denominan de orden p+q, siendo su fórmula:

$$U(p,q) = \sum_x \sum_y (x - \bar{x})^p (y - \bar{y})^q f(x,y)$$

Donde I(x,y) es el valor de la intensidad del píxel en la posición x,y.

Multitouch: Técnica de interacción persona-computador. Esta tecnología consiste en una pantalla táctil que reconoce simultáneamente múltiples puntos de contacto y en un software asociado a ésta que permite interpretar dichas interacciones simultáneas.

Tabletop: Mesa usada para la interacción persona–ordenador. La mesa consiste en una superficie transparente en la que se depositan los objetos para ser identificados, un proyector para mostrar una imagen con la que interactuar a través de los objetos y una cámara (vídeo, web,...) para detectar los objetos encima de la superficie.

Umbralización: Acción que consiste en separar, en una imagen en escala de grises, el fondo del objeto, siempre y cuando el fondo y los objetos tengan sus niveles de gris agrupados en 2 niveles dominantes, estableciendo un umbral de separación que puede ser variable.

XML: Extensible Markup Language (Lenguaje de marcas extensible). Formato usado para expresar información estructurada a través de etiquetas de la manera más abstracta y reutilizable posible. Una etiqueta consiste en una marca hecha en el documento, que señala una porción de éste como un elemento.

B.4 Modelo funcional

Como ya se ha dicho, el modelo funcional se emplea para especificar el significado de las operaciones en el modelo de objetos. Para representar estas actividades se ha utilizado el diagrama de flujo de datos.

B.4.1 Diagrama de flujo de datos

A continuación se muestra el flujo de datos que circula entre los diferentes procesadores y elementos de Reactivision (ver fig 63-64-65-66-67).

Los elementos que se encuentran coloreados en rojo son los que se han modificado:

- Detector de fiduciales y *blobs* para el reconocimiento de los nuevos fiduciales y de los *blobs*

- Servidor TUIO para el envío de la nueva información.

Y los coloreados en amarillo son los creados para el desarrollo del proyecto:

- Servidor XML, para el envío de los ficheros de información XML.

DFD Nivel 0:

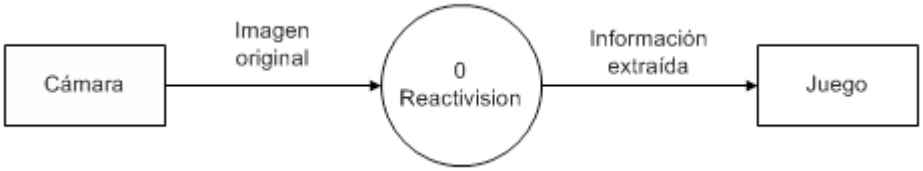


Figura 63 Diagrama de flujo de datos de Reactivision. Nivel 0.

DFD NIVEL 1:

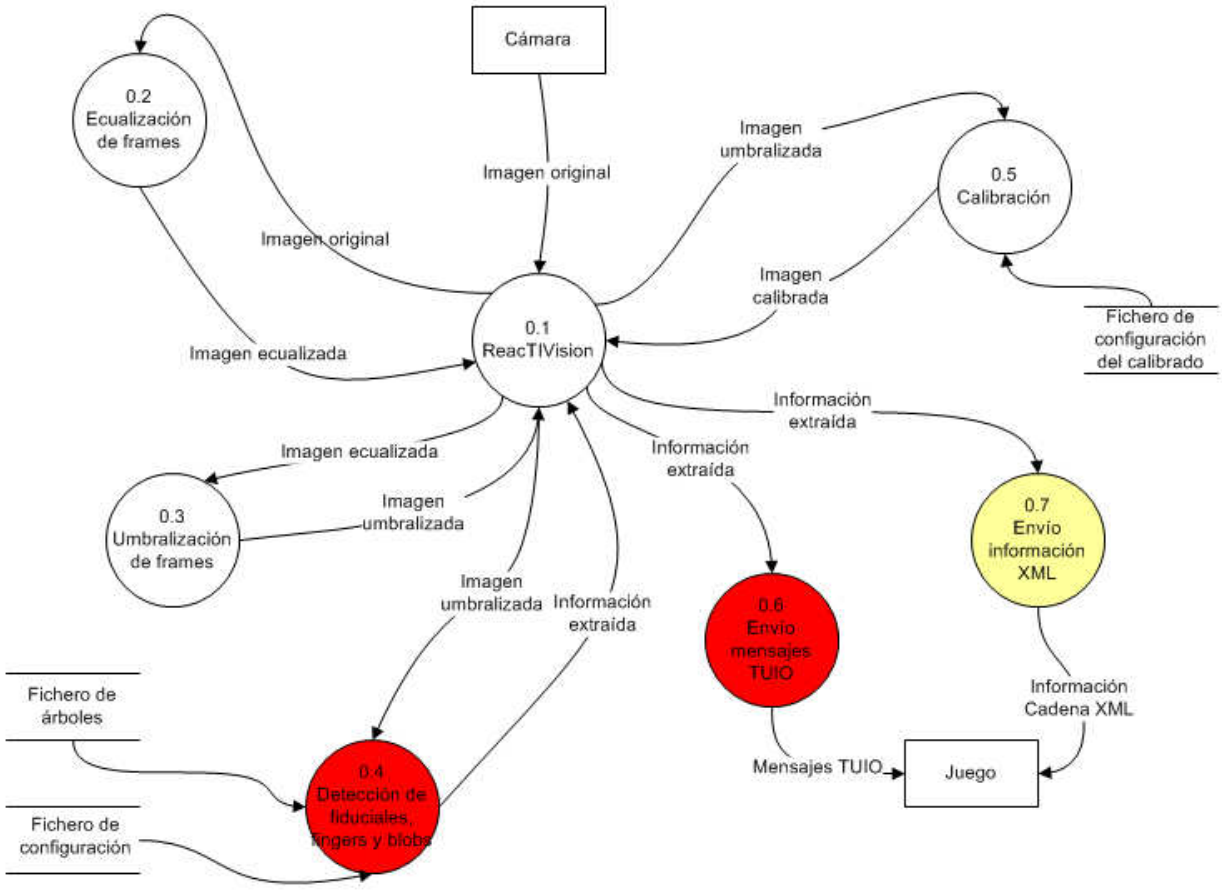


Figura 64. Diagrama de flujo de datos de Reactivision. Nivel 1.

DFD Nivel 2, detección de fiduciales, fingers y blobs:

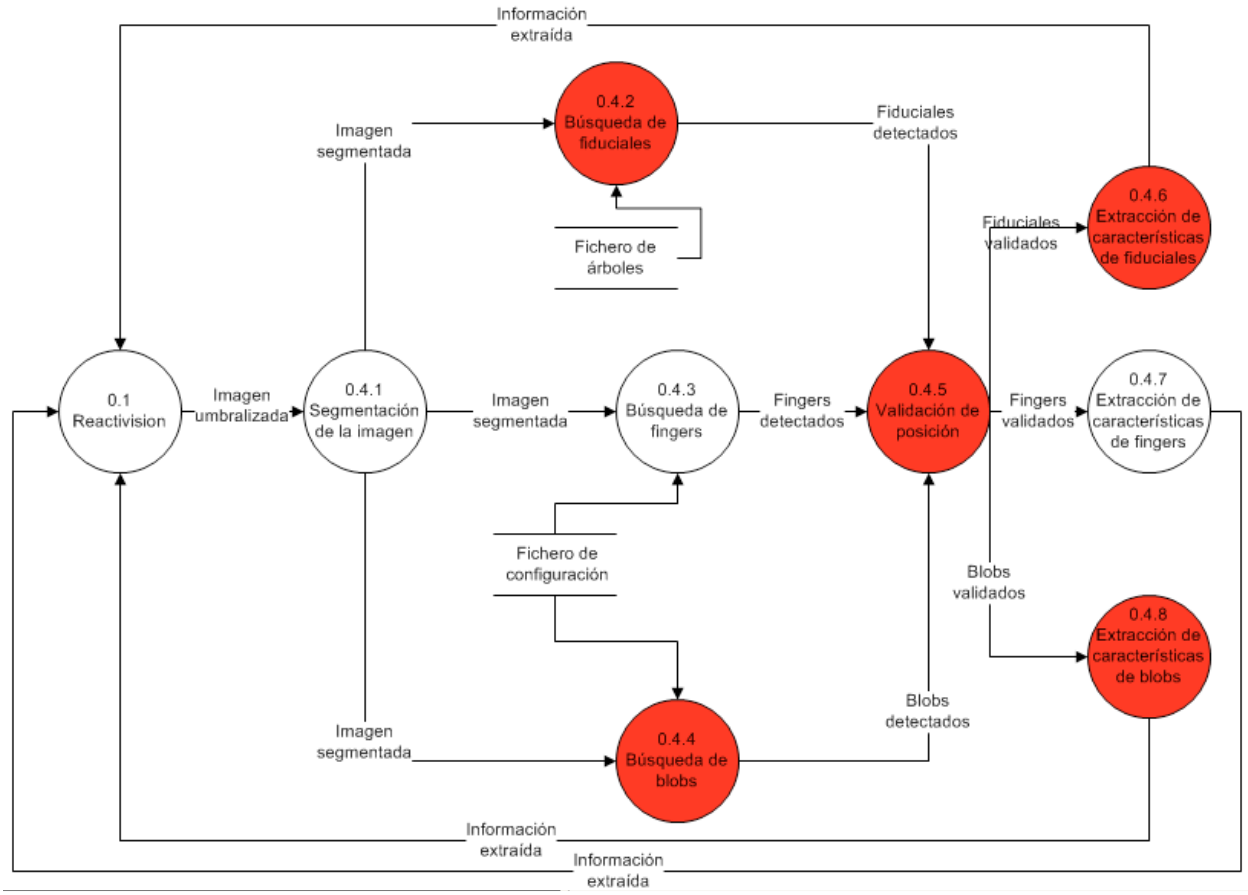


Figura 65. Diagrama de flujo de datos de Reactivision. Nivel 1, detección de fiduciales, *fingers* y *blobs*

La imagen umbralizada es segmentada, obteniendo las diferentes regiones que la forman. En la imagen segmentada se procede a la búsqueda de posibles *fiduciales*, *fingers* y *blobs* que posteriormente son validados para evitar falsos positivos o confusión entre objetos. Por último la información de cada uno de los objetos detectados es extraída.

DFD Nivel 2, envío de mensajes TUIO:

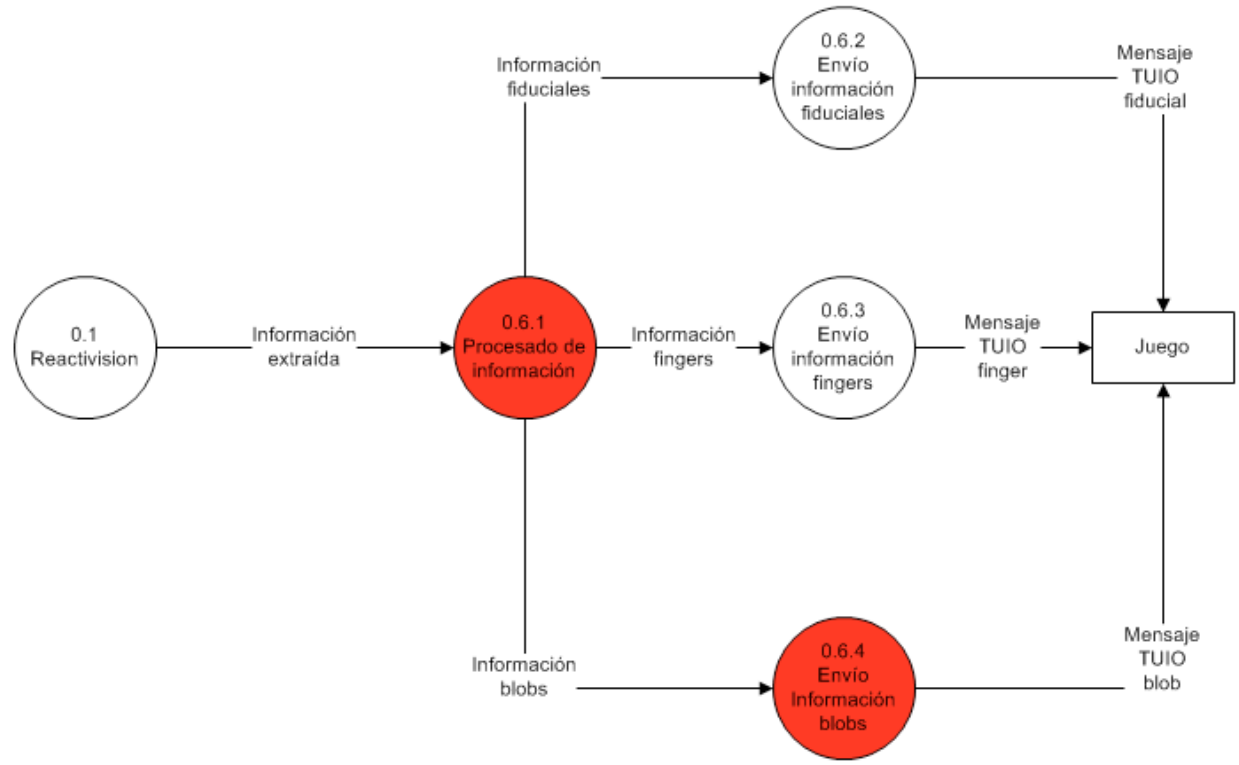


Figura 66. Diagrama de flujo de datos de Reactivision. Nivel 1, envío de mensajes TUIO

La información extraída de los diferentes objetos situados detectados es transmitida al servidor TUIO, que se encarga de procesarla y dividirla en información relativa a fiduciales, fingers y blobs. Por último los paquetes TUIO son creados y enviados a los juegos.

DFD Nivel 2, envío información XML:

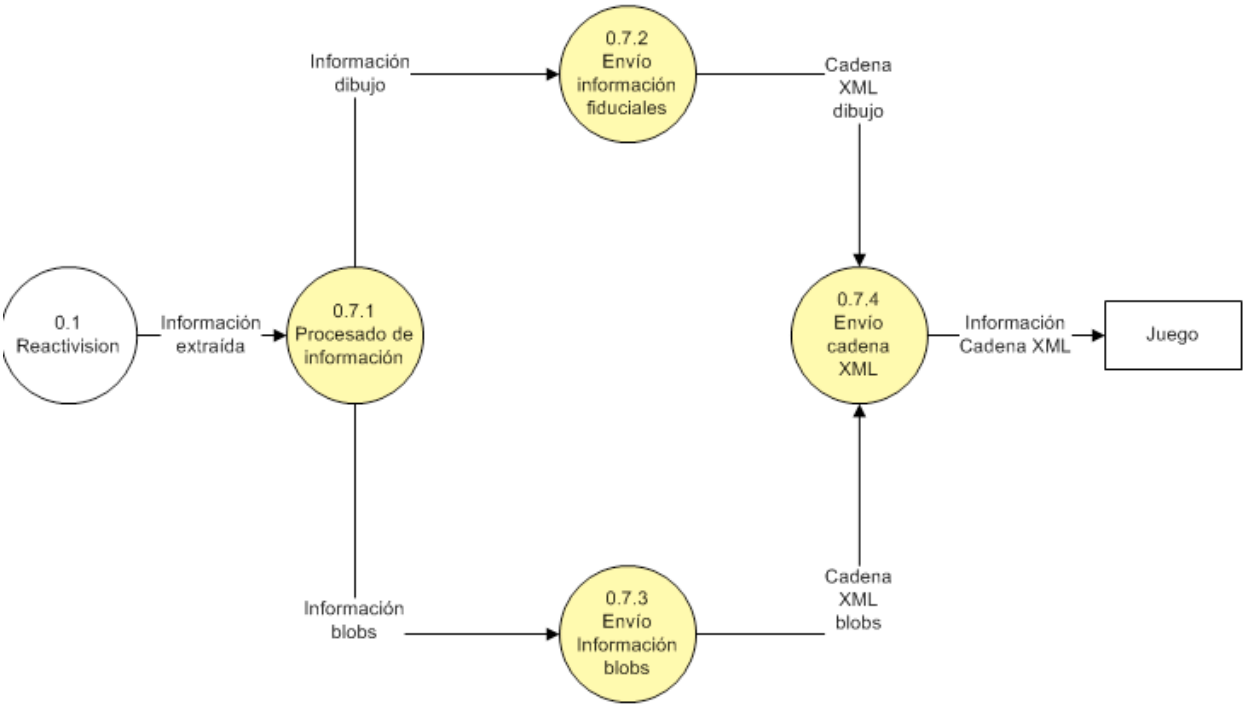


Figura 67. Diagrama de flujo de datos de Reactivision. Nivel 1, envío información XML

La información extraída de los blobs y fiduciales pertenecientes a dibujos, es transmitida al servidor XML, que se encarga de procesarla y generar a partir de ella la cadena XML. Por último la información es enviada a los juegos.

En la figura 68 se muestra el flujo de información entre el usuario y el interfaz gráfico. Los elementos que se encuentran en rojo son los que se han modificado para mostrar nueva información debido a las modificaciones. El detector de fiduciales y de *blobs* se ha modificado para que muestre por el *display* los identificadores de los nuevos fiduciales, las opciones de configuración del tamaño del folio para la captura de dibujos, así las opciones de configuración de los parámetros de los nuevos fiduciales.

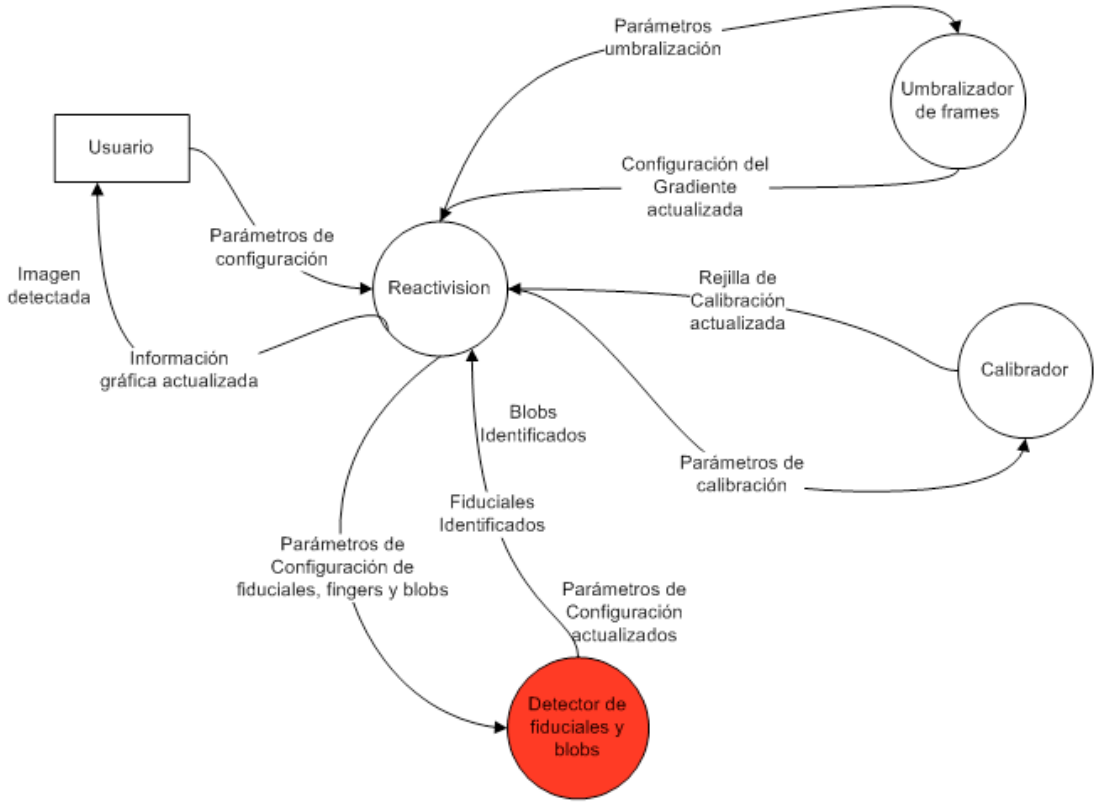


Figura 68 Diagrama de flujo de datos de la interfaz de Reactivision

Anexo C. Alternativas estudiadas

En esta sección se explica las diferentes alternativas estudiadas de los algoritmos empleados en las distintas partes del proyecto, analizando sus ventajas y desventajas y porque finalmente fueron descartadas.

C.1 Nueva colección de fiduciales

A continuación se detalla en qué consisten los fiduciales de Reactivision lo que permitirá comprender mejor las limitaciones que estos conllevan, así las diferentes alternativas de reconocimiento de los nuevos fiduciales estudiadas y finalmente descartadas.

C.1.1 ¿Qué son los fiduciales?

En primer lugar se estudió en detalle los fiduciales estándar de Reactivision para poder entender las limitaciones que estos suponen. Los fiduciales son marcadores impresos que se colocan en la base de los juguetes para que puedan ser reconocidos (figura 69). El diseño de los fiduciales suele ser sencillo debido a la necesidad de que un ordenador pueda detectarlos de manera rápida. Por otra parte esta sencillez permite que pequeñas variaciones en el diseño puedan generar un gran número de fiduciales. Los fiduciales utilizados son siempre en blanco y negro, ya que como se ha indicado se utiliza luz infrarroja para iluminar el prototipo y si los fiduciales fueran de colores, no podrían ser reconocidos.

Figura 69 Juguetes identificados con fiduciales.

Los fiduciales que Reactivision interpreta tienen forma de ameba, ya que han sido diseñados para optimizar la distancia entre los contornos negros y blancos para que la cámara los detecte bien, usando círculos, que pueden ser concéntricos. Estos círculos proporcionan una orientación que puede ser usada en el procesado de la información [CH09]. Este diseño en

ameba obliga a que los fiduciales cuanto más pequeños sean tengan que tener una forma más o menos cuadrada, lo que limita el tipo de juguetes a utilizar o condiciona la estética del juguete. A su vez para el correcto reconocimiento de los fiduciales, Reactivision obliga a que estos tengan unos tamaños mínimos, volviendo al problema del tipo de juguetes a utilizar. En la figura 70 se muestra un ejemplo de fiducial reconocido por Reactivision. Si este tamaño es menor se producen errores y conflictos en la identificación de fiduciales.

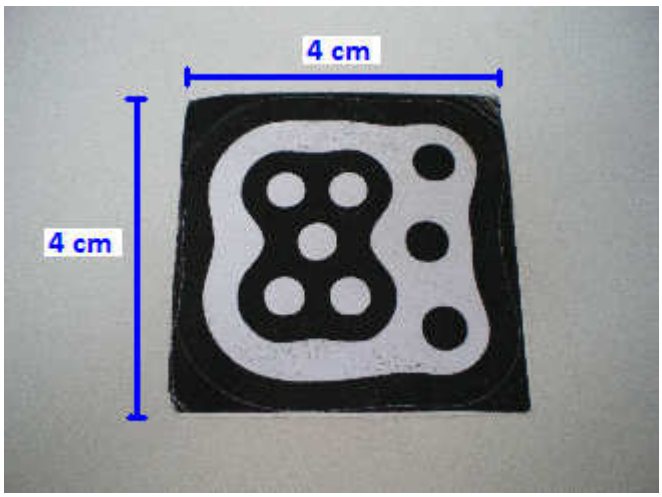


Figura 70 Fiducial reconocido por Reactivision.

C.1.2 Representación de los fiduciales

La información que distingue a un fiducial de otro viene representada por un árbol que contiene la estructura de las zonas en blanco y negro que hay en él. La construcción del árbol se realiza tomando como nodo raíz el color más externo del fiducial, después se saca el número de zonas de color opuesto que hay dentro de la zona inicial. Para cada nueva zona detectada se vuelve a realizar el paso anterior, y así hasta que todas las zonas del fiducial se han analizado. En la figura 71 se observa un fiducial y la representación correspondiente en forma de árbol.

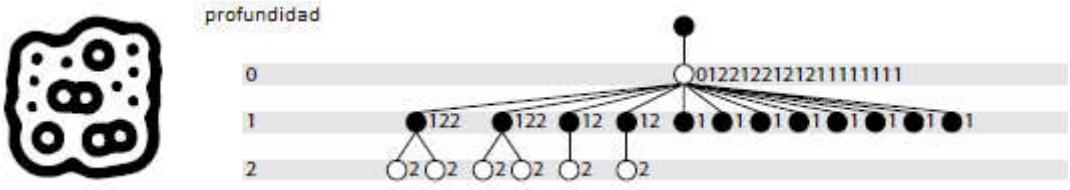


Figura 71 Fiducial y su árbol correspondiente

Para evitar datos irrelevantes, como por ejemplo el color del fondo, se recurre a un proceso de minimización de los árboles consistente en la eliminación las partes comunes, dejando únicamente las partes que difieren entre unos y otros. La representación óptima se realiza con 3 o más niveles de profundidad para tener suficiente robustez, ya que, usando

menos, se pueden producir falsas identificaciones en la imagen captada, como considerarlo un fiducial distinto o no detectarlo.

Analizando el ejemplo de la figura 71 se puede eliminar el nodo raíz de color negro ya que la parte importante se encuentra en los nodos inferiores. El contorno del fiducial no aporta ninguna información sobre la diferenciación de éste con respecto a otro. Las zonas interiores son las que pueden variar y, por lo tanto, dar esa distinción. En la Figura 72 se observa cómo queda el árbol después de aplicarle una simplificación.

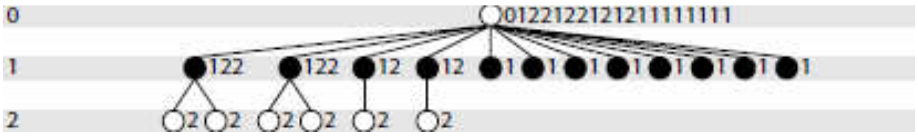


Figura 72 Árbol reducido del fiducial de la figura 67

Por otra parte el cálculo de la orientación puede ser o no ser necesario dependiendo del uso que se le vaya a dar al fiducial. Este cálculo se realiza a partir de los círculos de menor tamaño del fiducial tal como se describe a continuación: se calcula el centro de la unión de todos los círculos del mismo color, obteniendo dos centros, uno para los blancos y otro para los negros. Cuando se han obtenido ambos centros se calcula el vector orientación que va desde el centro blanco hacia el centro negro, tomando como 0º el vector apuntando hacia arriba y aumentando con el giro en el sentido de las agujas del reloj.

C.1.3 Nuevos diseños de fiducial

Una vez analizada y comprendida la estructura que poseía un fiducial de Reactivision, se procedió al estudio del sistema de reconocimiento que se empleaba para su posterior adaptación al reconocimiento de tamaños más pequeños.

Los fiduciales de Reactivision son identificados únicamente por su estructura topológica. Esta topología es representada mediante una estructura en árbol como se ha explicado anteriormente. El número de nodos en un árbol y su profundidad afecta directamente al tamaño mínimo que debe tener un fiducial. Además, los árboles más pequeños permiten un menor número de posibilidades de representación, dando lugar a una mayor probabilidad de encontrar detecciones falsas cuando se trata de reconocerlos en una escena (Bencina, Ross et al), de ahí su limitación en cuanto a tamaño mínimo.

El sistema de reconocimiento comienza con una segmentación de la imagen en regiones. Esta operación construye un grafo de adyacencia de regiones a partir de la imagen umbralizada. El algoritmo recorre línea a línea la imagen creando nuevas regiones o actualizando las ya existentes. Se considera una región, a un conjunto de píxeles del mismo color y que se encuentran juntos, es decir comparten un vértice o una arista. Las adyacencias de cada región con otras se van guardando en un vector, sin embargo cuando este vector se llena, se deja de estudiar esa región y se pasa a otra. Como resultado este algoritmo construye

un grafo de adyacencia incompleto A partir de él Reactivision procede a la identificación de fiduciales. Primero busca un posible candidato, estos candidatos son elegidos en base a una serie de estadísticas precomputadas sobre las características de los árboles que forman el diccionario de fiduciales utilizado. Estas características son por ejemplo el total de nodos detectados o la profundidad máxima de las hojas. Después se busca una correspondencia en los grafos con alguno de los árboles del diccionario. Es evidente que esta incompletitud en el grafo aumenta considerablemente las posibilidades de falsos positivos.

Para solucionar el problema de mala detección de fiduciales, se **modificó ese algoritmo de reconocimiento para que completase los grafos de adyacencia** y así aumentase sus probabilidades de acierto a la hora de buscar correspondencia con los fiduciales del diccionario. Tras probar este nuevo sistema con fiduciales más pequeños, se comprobó que el reconocimiento no había mejorado. Esto era debido a que la imagen capturada por la cámara no tenía la suficiente resolución, lo que provocaba que los fiduciales no se distinguiesen bien y unas áreas se confundiesen con otras.

Tras comprobar que el mayor problema se encontraba en el hardware y no en el sistema de reconocimiento, ya que el mal reconocimiento era debido a la mala resolución de la cámara utilizada, se decidió **diseñar una nueva colección de fiduciales** más sencillos y los algoritmos para su detección que no dependiesen de la resolución de la cámara para su reconocimiento.

C.1.4 Estudio de otros algoritmos

El primer algoritmo que se estudió, fue el de **recubrimiento convexo**, intenta describir la forma de un objeto y la define como la forma encerrada en una banda elástica situada alrededor del objeto en cuestión. La forma que hay que añadir a un objeto determinado para crear su recubrimiento convexo se denomina deficiencia convexa [Sc98].

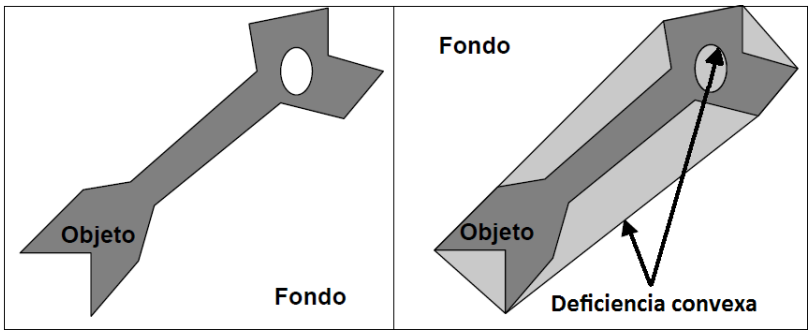


Figura 73 Objeto y su recubrimiento convexo.

En este algoritmo la detección de agujeros consiste en la búsqueda de la parte de la deficiencia convexa que no es adyacente al fondo. Ya que como se observa en la figura 73 ésta corresponde con los agujeros del objeto. Una de las posibilidades para encontrar el recubrimiento convexo de un objeto, es ir recorriendo este en grupos de 3x3 píxeles y poner a 1 el pixel central de las zonas que representan una concavidad y volver a repetir este paso

hasta que ya estén todas las concavidades rellenas, de esta manera las zonas marcadas con unos corresponderán con el recubrimiento convexo y el resto a el exterior a este.

Tras analizar el funcionamiento de este algoritmo se decidió finalmente implementar el **algoritmo de conectividad entre regiones**, explicado en el capítulo 2, como solución a la detección de los nuevos fiduciales. Aunque los resultados de ambos algoritmos son similares, se escogió éste algoritmo y no el de recubrimiento convexo dado que este último era notablemente más complejo de implementar, primero se calcula la deficiencia convexa y después la conectividad entre áreas, tanto para rellenar las zonas cóncavas como para identificar los agujeros, mientras que en el algoritmo de Moore solo se calcula la conectividad entre áreas para después buscar los agujeros.

C.2 Extracción del contorno en objetos planos

A continuación se detallan los diferentes algoritmos estudiados para la extracción de características de objetos planos explicada en el capítulo 3.

C.2.1 Estudio de otros algoritmos

C.2.1.1 Extracción de píxeles frontera

Para la primera fase en la extracción del contorno de un objeto, es decir, la extracción de los píxeles frontera, se estudiaron diferentes algoritmos todos ellos basados en la conectividad entre los píxeles.

En primer lugar se estudió el **algoritmo de extracción del contorno 'cuadrado'** que consiste en: dado una imagen en la que el objeto es totalmente de un color, por ejemplo negro y el fondo es totalmente blanco, se debe encontrar un pixel negro que se denominara pixel de comienzo y que limitará con la zona blanca. Este pixel puede ser encontrado de muchas maneras, como por ejemplo recorrer la imagen por filas y quedarse con el primer pixel de color negro que se encuentre. Una vez encontrado este pixel de comienzo se deben seguir una serie de pasos: siempre que se encuentre con un pixel negro, es decir perteneciente al objeto, se debe girar a la derecha, o bien si el pixel es blanco siempre se debe girar a la izquierda. Estos pasos se repiten hasta que el volvamos a encontrar el pixel de comienzo. Los píxeles negros que se hayan ido encontrando a lo largo del algoritmo serán los que formen el contorno el objeto. Lo importante en este algoritmo es el "sentido de la orientación". Los giros a la izquierda y la derecha se hacen deben ser realizados con respecto a la posición actual, que depende de la forma en que se entró en el pixel. Por lo tanto, es importante no perder de vista la orientación actual con el fin de hacer los movimientos correctos. Para entender mejor el algoritmo en la figura 74 se puede ver un ejemplo de los primeros pasos seguidos en el recorrido hasta encontrar todos los píxeles del contorno.

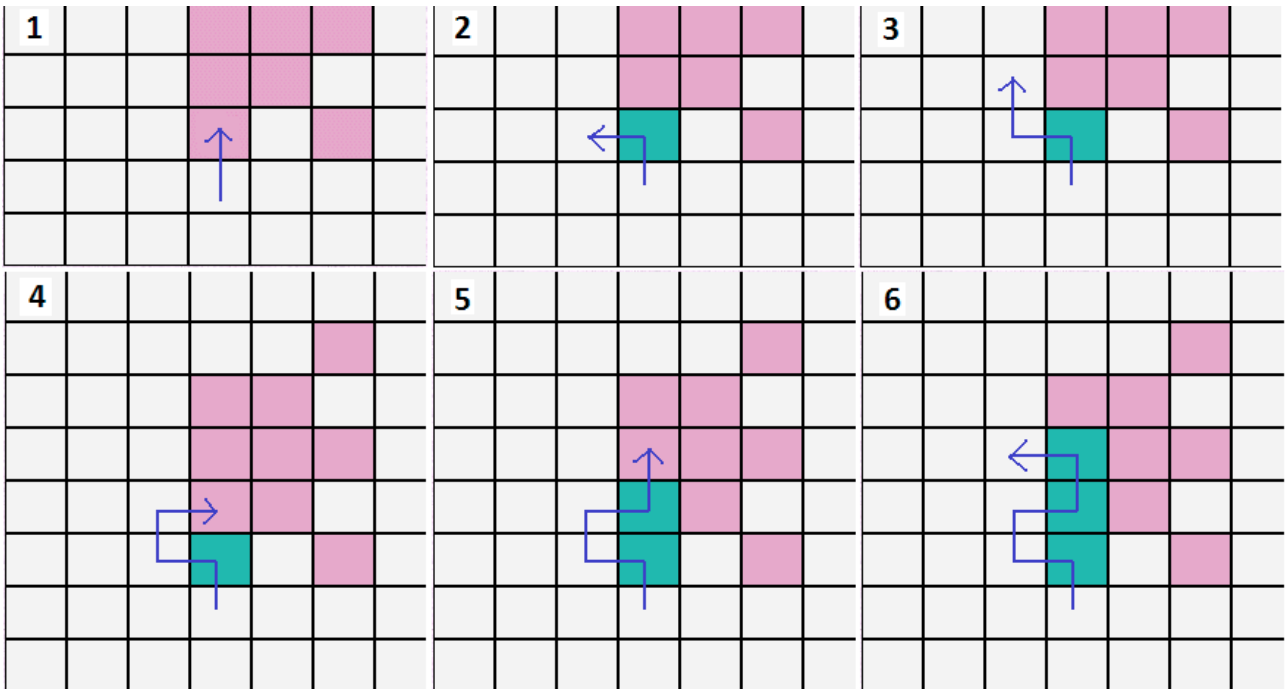


Figura 74 Ejemplo de los primeros pasos seguidos en el recorrido hasta encontrar todos los píxeles del contorno

Este algoritmo no satisface las necesidades de este proyecto ya que una vez implementado se detectó que la mayor parte de las veces el contorno no era definido de forma correcta. Tras estudiar el algoritmo en profundidad se detectó que a través de él solo se pueden encontrar los contorno de aquellos objetos cuyos píxeles sigan un patrón de conexión 4-vecino, es decir los píxeles pegados unos con otros deben seguir la estructura de la figura 75, de lo contrario se pueden perder la mayoría de los píxeles que forman el objeto.

En el caso de los objetos que pueden aparecer en las imágenes tratadas no se asegura que los píxeles que lo componen cumplan la propiedad de 4-vecindad, por lo que este algoritmo fue descartado. En la figura 76 puede verse un ejemplo en el que el algoritmo fallaría.

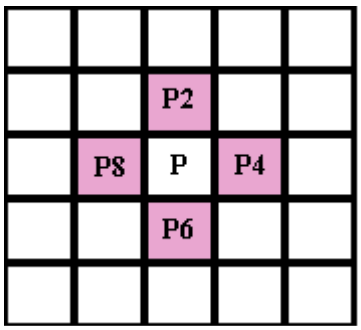


Figura 75 Relación 4-vecinos

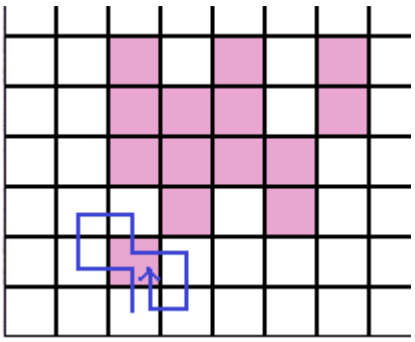


Figura 76 Contorno mal tratado por el algoritmo de extracción de contorno 'cuadrado'

Otro algoritmo estudiado fue el de **Theo Pavlidis**, que consta una vez el encontrado el pixel de comienzo de tres: siempre hay que fijarse en los tres píxeles que estén delante del actual, si el pixel de delante a la izquierda es negro ir a él (figura 77)

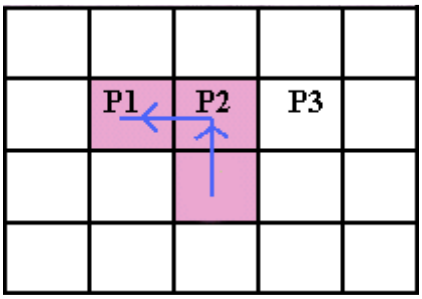


Figura 77 Paso a seguir si el pixel de delante a la izquierda al actual es negro., en el algoritmo de Theo Pavlidis.

En el caso de que este pixel sea blanco ir a P2 y si este es también blanco ir a P3 siguiendo el paso de la figura 78.

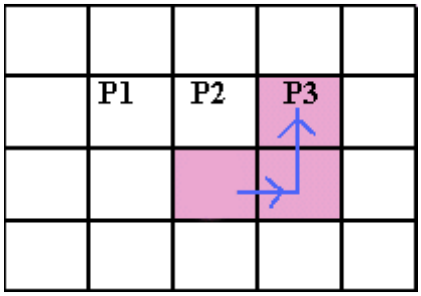


Figura 78 Paso a seguir si Los píxeles P1 y P2 son blancos y P3 es negro, en el algoritmo de Theo Pavlidis.

Si ninguno de estos tres pasos lleva a un pixel negro, rotar 90º y probar. Si tras 3 intentos de giro no se llega a un pixel negro el algoritmo acaba. Lo más importante del algoritmo es tener en cuenta la orientación en la que se encuentra en cada momento, para realizar los movimientos correctos, al igual que en el primer algoritmo tratado.

Pese a tratarse de un algoritmo más complejo, tras su implementación se comprobó que este algoritmo también presentaba problemas. El algoritmo funciona bien si la estructura del objeto a tratar cumplía una conectividad 4-vecina, sin embargo podía fallar si se trataba de una

conectividad 8-vecina pero no 4 vecina. En la figura 79 se muestra como se vuelve al pixel de comienzo sin haber encontrado todos los píxeles del contorno.

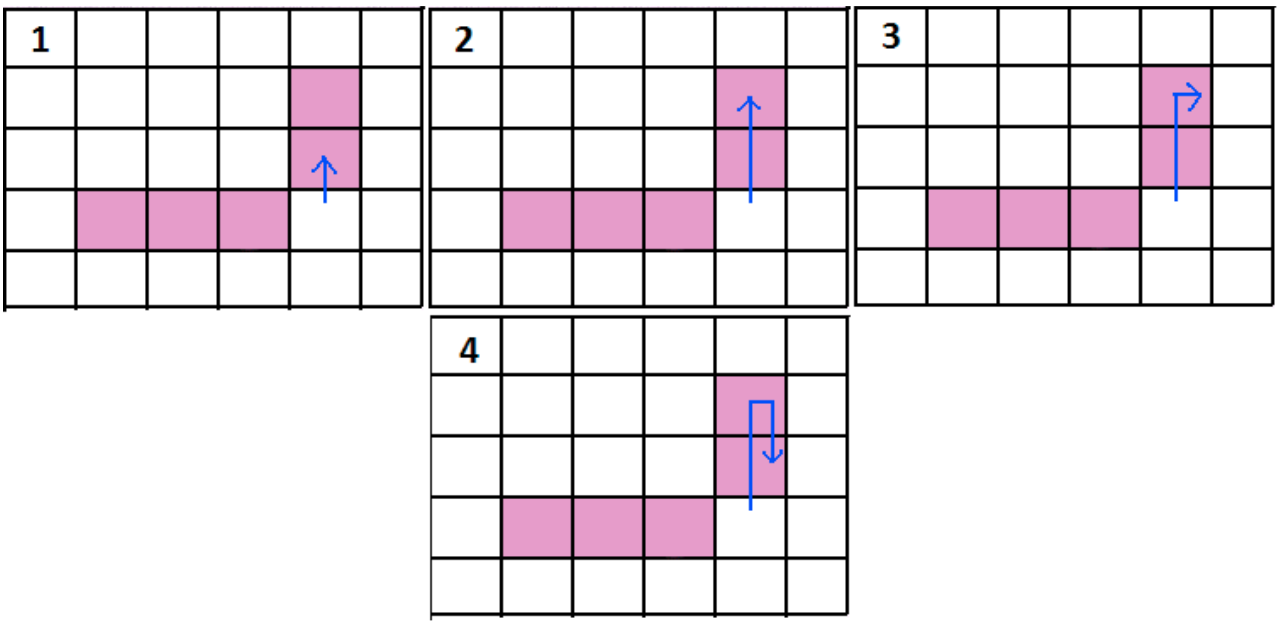


Figura 79 Tratamiento erróneo del contorno por el algoritmo de *Theo Pavlidis*

En la figura 80 se observa como este algoritmo supera sin problema las limitaciones del algoritmo anteriormente explicado, encontrando todos los píxeles que forman el contorno del objeto y por tanto demostrando que con conexión 4-vecina también funciona.

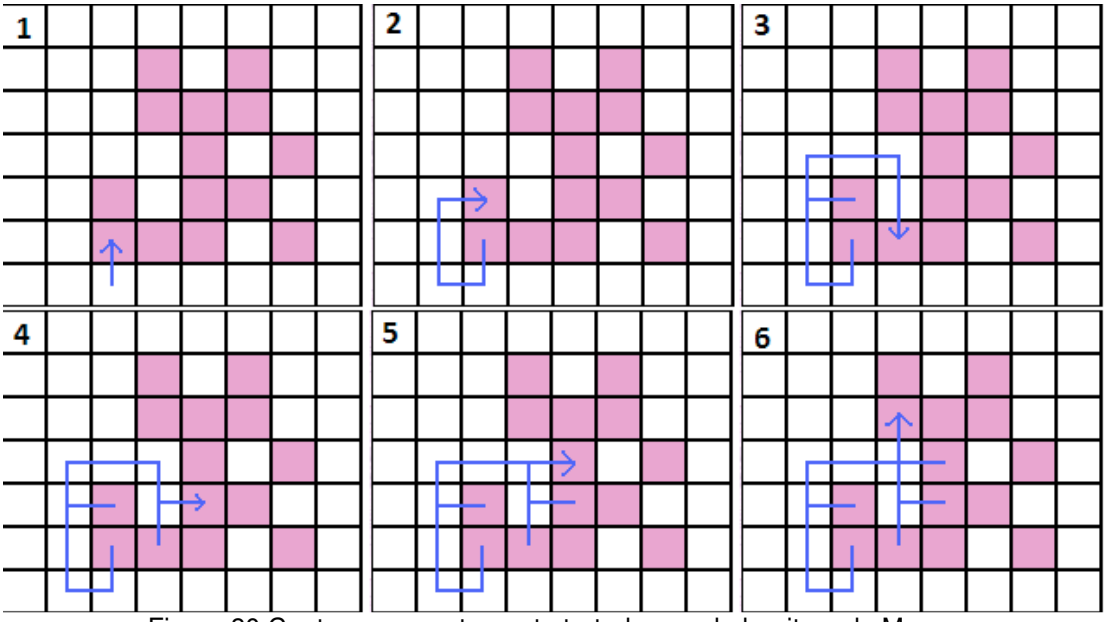


Figura 80 Contorno correctamente tratado por el algoritmo de Moore

Por los errores detectados en los algoritmos se decidió finalmente implementar el **algoritmo de Moore**, explicado en el capítulo 3, el único que no presentó ningún problema durante sus pruebas.

C.2.1.1 Construcción de segmentos del contorno

Para la segunda fase, es decir, la elección de los píxeles que finalmente forman el contorno y la construcción de los segmentos de éste, se estudiaron varios algoritmos basados en filosofías distintas. La primera filosofía estudiada fue la de aproximación poligonal, esta consiste en aproximar un contorno mediante el uso de un polígono. La aproximación será más exacta cuanto mayor sea el número de segmentos del polígono utilizado. El objetivo de esta técnica consiste en emplear el menor número posible de segmentos de tal manera que no se pierdan características esenciales del contorno. El primer algoritmo dentro de esta filosofía estudiado fue el **algoritmo de fusión**, se trata de ir ajustando los píxeles del contorno mediante una recta hasta que el error cometido en el ajuste supere un umbral preestablecido. Entonces, se almacenan los parámetros de la recta obtenida y se repite el proceso con los píxeles siguientes del contorno hasta que se hayan tratado todos. Al terminar el proceso, las intersecciones de las rectas adyacentes son los vértices del polígono (figura 81).

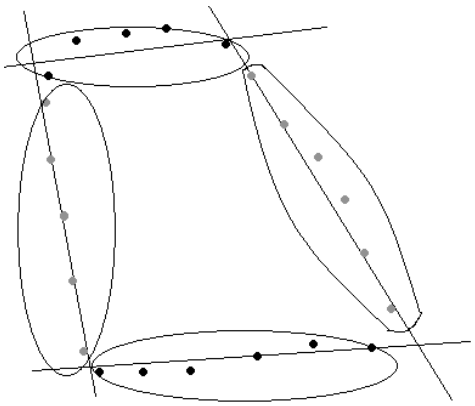


Figura 81 Ejemplo de aproximación mediante la técnica de fusión

El principal problema que presenta esta técnica es que los vértices generados no suelen corresponder con los vértices reales del contorno en especial con los que forman las esquinas.

El segundo algoritmo de este tipo estudiado fue la de **división recursiva**, esta consiste en dividir sucesivamente un segmento en dos hasta que se satisfaga un determinado criterio, como por ejemplo, que la distancia de los puntos de un tramo del contorno al segmento que los aproxima no sobrepase una distancia determinada. En caso de que la sobrepasen, el punto más lejano es considerado nuevo vértice donde se subdivide el segmento en dos. A continuación se muestra un ejemplo del proceso seguido (figura 82).

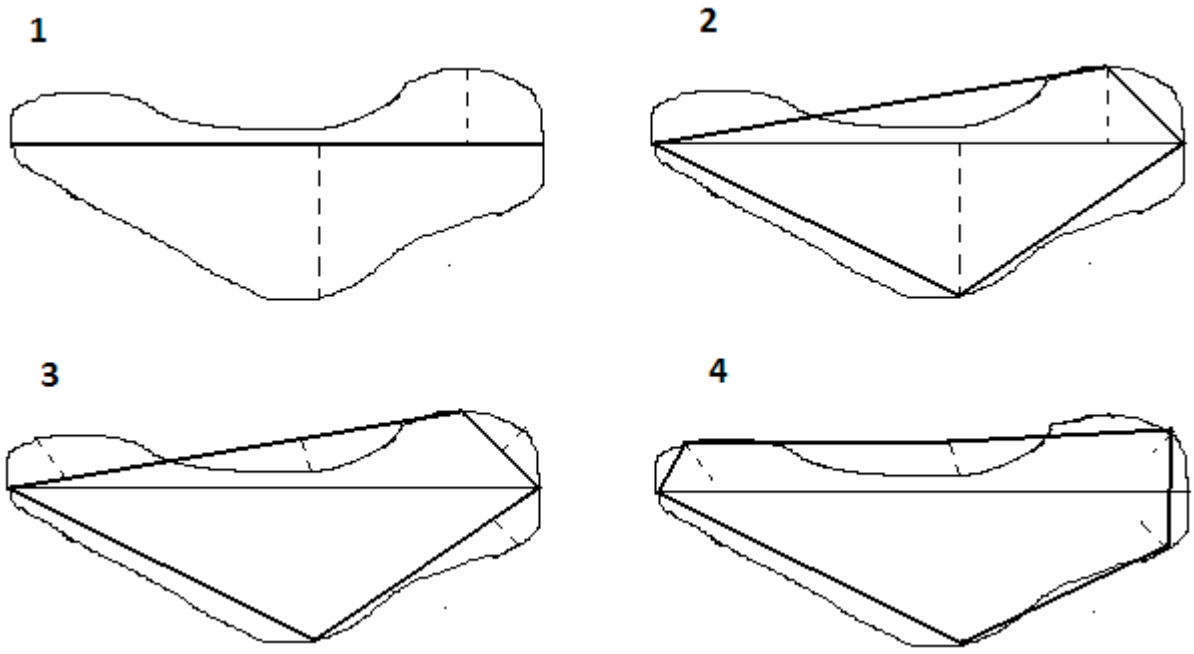


Figura 82 Proceso seguido en el método de división recursiva.

Este método tiene la ventaja de que detecta bien las esquinas del contorno, sin embargo tiene una alta sensibilidad al ruido en la imagen y que si el contorno contiene concavidades el proceso se hace más complicado.

Otro tipo de algoritmos estudiados fueron los de **representación polar**. La idea es representar el contorno como una función polar, para ello se calcula un punto característico del interior del contorno, por ejemplo el centro de masas, y a partir de él representar la distancia de cada punto del contorno a dicho centro. Este método posee el inconveniente de ser muy sensible a la posición del centro, de esta manera cualquier error o perturbación en el centro se refleja en el cálculo del contorno.

Finalmente se estudiaron algoritmos de **códigos de cadena**, estos se usan para representar la frontera en base a un conjunto de segmentos, de una longitud y dirección específica, conectados entre sí. Por lo general, esta representación está basada en segmentos de 8 direcciones. El procedimiento consiste en asignar una rejilla a la imagen y asignar a cada pixel de la frontera el nodo de la rejilla que se encuentre más cerca de él. Finalmente a partir de estos nodos, se crea un código que indica la dirección de los segmentos que forman el contorno. En la figura (83) se puede ver un ejemplo de los diferentes pasos seguidos en la técnica de cadena.

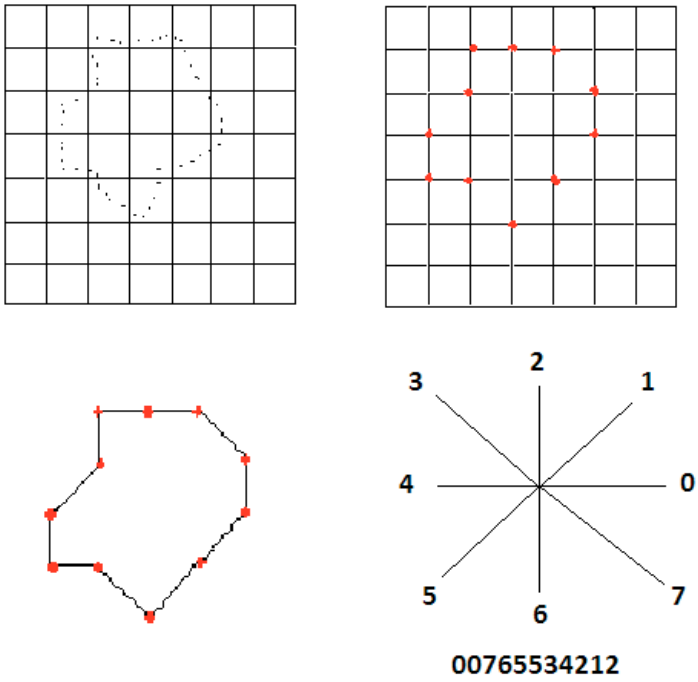


Figura 83 Proceso seguido en el algoritmo de cadena.

Debido a los problemas que presentaban cada uno de los algoritmos así como su dificultad de implementación. Se decidió utilizar finalmente una simplificación de la técnica de cadena, en la que solo se usa la primera parte de esta, es decir la obtención de los nodos. La formación de los segmentos fue diseñada por completo en este PFC (capítulo 3).

C.3 Protocolo de comunicación

Para superar las limitaciones de comunicación expuestas en el capítulo 4, en un principio se estudió la utilización de memoria compartida entre Reactivision y los juegos en Flash. Esta técnica haría que la velocidad se incrementase considerablemente, ya que ambos programas estarían trabajando sobre el mismo espacio de memoria y podrán evitarse los problemas de bloqueo con técnicas de exclusión mutua. Se procedió al estudio de las herramientas disponibles para la implementación de esta solución. Se partía del problema de que cada uno de las dos partes estaba implementada en lenguajes diferentes, Reactivision en C++ y los juegos en ActionScript (Flash). Tras buscar información sobre memoria compartida, se encontró que existía una API para C sobre Windows (Windows Api) que permitía el trabajo con memoria compartida. Esta API permitía crear un espacio de memoria compartida por varios procesos que podían acceder a ella de manera excluyente, es decir para que no hubiese problemas de accesos o condiciones de carrera, los diferentes procesos que quisiesen acceder a este espacio de memoria, deberían utilizar mecanismos de sincronización como semáforos, eventos o algún mecanismo de exclusión mutua para que dos procesos no puedan acceder a la vez al espacio de memoria compartida.

Sin embargo no se encontró ninguna API o recurso para utilizar memoria compartida con ActionScript, por lo que finalmente se decidió buscar otras alternativas, finalmente implementando el protocolo basado en sockets y XML, explicado en el capítulo 4.

Anexo D. Otros frameworks y tabletops existentes.

D.1 Frameworks

A continuación se analizan las diferencias que presentan algunos *frameworks* existentes: Community Core Vision y Touchlib respecto a Reactivision.

D.1.1 Community Core Vision

Community Core Vision o CCV es un *framework* de código abierto de visión por computador desarrollado por el NUI Group [NUI, web]. Como Reactivision toma imágenes con una cámara y las analiza para detectar objetos en la superficie de un *tabletop*. A diferencia del *framework* utilizado en este proyecto, CCV se centra en la detección de eventos táctiles como el tocar la superficie o deslizar los dedos por encima del *tabletop*. También implementa el reconocimiento de objetos sin forma definida así como el reconocimiento de objetos identificados con fiduciales, sin embargo este tipo de reconocimiento no está todavía suficientemente desarrollado. Al igual que Reactivision puede comunicar los datos extraídos a otras aplicaciones a través de TUIO.

Como se puede ver en la figura 84 la ventaja que CCV ofrece frente a Reactivision es la cuidada interfaz de usuario de la que viene provista, siendo esta la única ventaja frente a Reactivision.

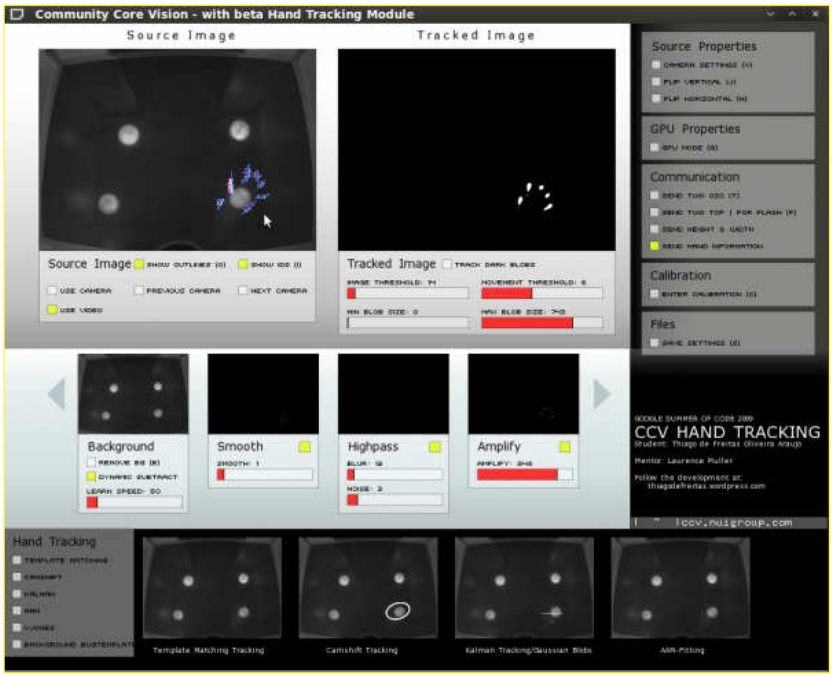


Figura 84 Interfaz de usuario de CCV

D.1.2 Touchlib

Touchlib es una librería para aplicaciones multitáctiles que usen iluminación difusa [Touchlib, web]. Provee de detección y seguimiento de objetos planos sin ningún tipo de marcador. También provee funcionalidades limitadas de reconocimiento de fiduciales, pero no es su punto fuerte. A diferencia de Reactivision, sólo envía la información obtenida a aplicaciones programadas en C++. En la figura 85 se puede observar como la interfaz es muy similar al de Reactivision.

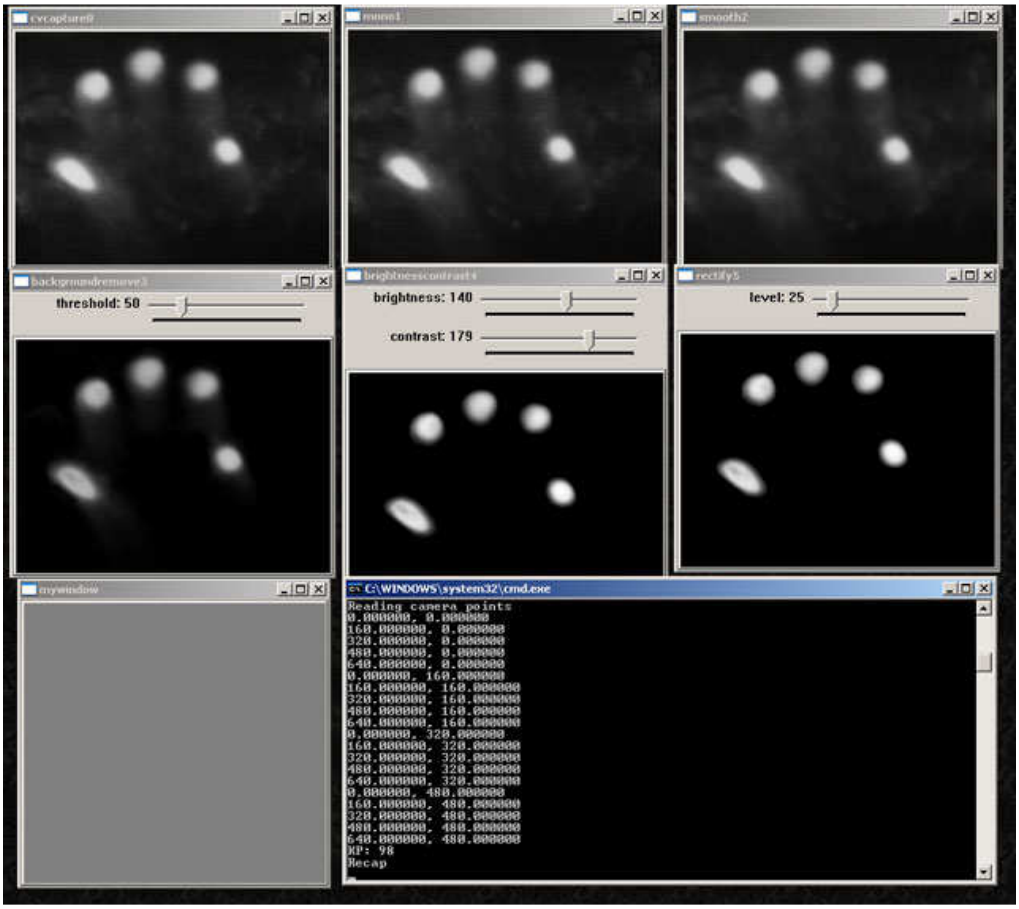


Figura 85 Interfaz gráfica de touchlib.

La utilización de cualquiera de estos dos *frameworks* en este PFC supondría el tener que volver a implementar funcionalidades como sería el reconocimiento de objetos planos en CCV y el reconocimiento de fiduciales en Touchlib.

D.2 Tabletops

A continuación se analizan algunos de los *tabletops* disponibles en el mercado y en trabajos de investigación, realizando una comparación con las funcionalidades desarrolladas en este PFC

D.2.1 Reactable

Este *tabletop* [Reactable, web] es utilizado a modo de un instrumento musical electrónico colaborativo dotado de una interfaz tangible. Está inspirado en los sintetizadores modulares de los años sesenta. Fue desarrollado por el Grupo de Tecnología Musical de la Universidad Pompeu Fabra de Barcelona. Sus desarrolladores son los mismos que desarrollaron Reactivision. Cada fiducial, lleva asociado un sonido y cuando se coloca uno de estos fiduciales en el *tabletop* y Reactivision lo reconoce se emite el sonido. Además la orientación y distancia entre los diferentes objetos, permite variar la frecuencia a la que suena la música.

La principal ventaja que presenta NIKvision sobre este *tabletop* reside en los diferentes juegos que implementa, ya que Reactable solo está destinado a la creación de música.

En la figura 86 se muestra el *tabletop* en funcionamiento y la variación de frecuencia mediante la orientación y distancia de los objetos

Figura 86 Parte superior del *tabletop* Reactable y como dependiendo de la distancia y rotación de los objetos se producen sonidos de diferentes frecuencia*s.*

D.2.2 Flux Digital Tabletop

Tabletop con fines gráficos y de diseño de planos, que sirve de pizarra digital. Está diseñado para poder hacer trazos de gran precisión y por ello permite la utilización de lápices especiales y de diferentes tamaños, pudiéndose también ampliar la imagen. A su vez permite el manejo por parte de varias personas de forma simultánea. Cuando se sitúa la punta de uno de los lápices sobre la superficie del *tabletop,* éste la reconoce y va dibujando una línea, siguiendo el recorrido que se haga [LSPHBDP09].

Como puede verse en la figura 87 este *tabletop* está diseñado para el dibujo de precisión, como puede ser el dibujo de planos de arquitectura. En el *tabletop* usado durante el proyecto también permite el dibujo sobre la superficie, además permite el dibujo con pinceles

convencionales y el grosor del trazo dependerá de la presión que se ejerza con estos sobre la superficie de la mesa, lo que ofrece una mayor naturalidad en su manejo.

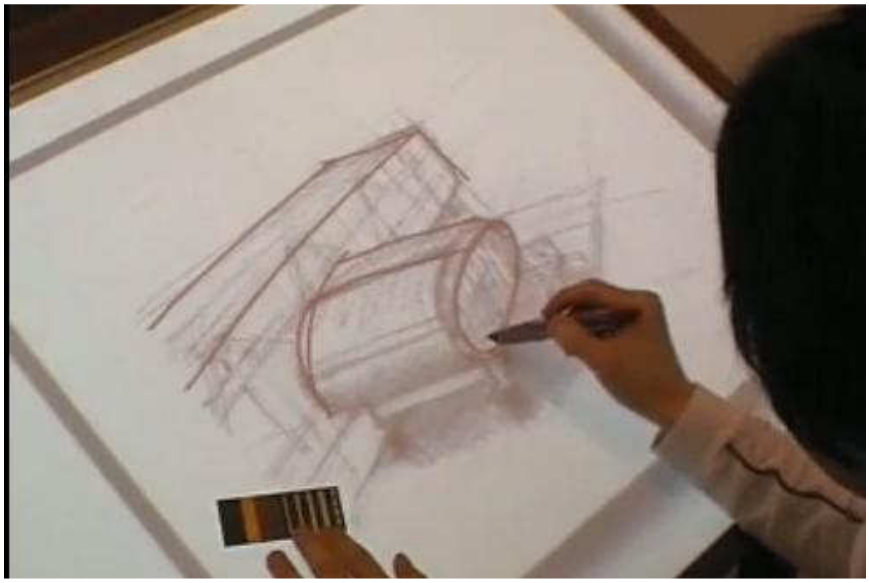


Figura 87 Dibujo realizado en el *tabletop* Flux Digital

D.2.3 U-Touch 103" Multitouch Air Hockey

Tabletop que simula el clásico juego de *Air Hockey*, que se puede encontrar en los salones recreativos [U-Touch, web]. La mesa detecta el punto de contacto de la mano con la superficie del tablero, siguiendo su trayectoria cuando se desplaza. Posteriormente realizan un cálculo de trayectoria y de los rebotes del disco cuando es golpeado para mostrar el movimiento del disco como si fuese real. La interacción se realiza a través de la detección de los dedos, funcionalidad que ya viene implementada en Reactivision.

Este *tabletop* no presenta ninguna mejora respecto al utilizado en este proyecto. La detección de los dedos es la única función de reconocimiento y como ya se ha explicado en apartados anteriores, ésta ya viene implementada en Reactivision. En la figura 88 se muestra una imagen de la aplicación siendo usada por 2 personas.

Figura 88 *Tabletop U-Touch 103" Multitouch Air Hockey*

D.2.4 Lumisight Table

Tabletop diseñado para la interacción simultánea de varios usuarios que permite la representación gráfica en las cuatro direcciones de la mesa, permitiendo así una mejor visión a cada uno de los usuarios (figura 89). Su funcionamiento se basa en la detección de los dedos de las manos así como de objetos identificados a través de marcadores [KHLNM06]. Como se observa en la figura 90 estos marcadores, se limitan a 4, lo que a diferencia con lo implementado en este proyecto, limita en gran medida el número de objetos a utilizar. Con respecto a la direccionalidad de la representación gráfica, NIKvision puede conseguir resultados similares, ya que esto depende de los juegos implementados en Flash. Al igual que el resto de tabletops mostrados, Lumisight Table solo está destinado a una tarea y las funcionalidades que implementa pueden ser replicadas en Reactivision.

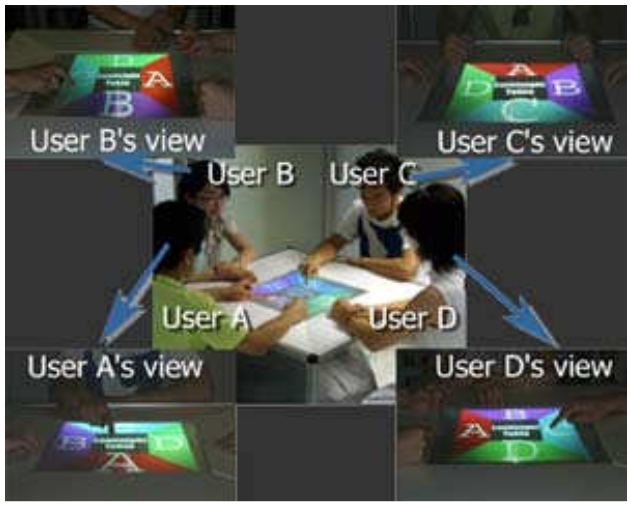


Figura 89 Interacción multi usuario en Lumisight table.

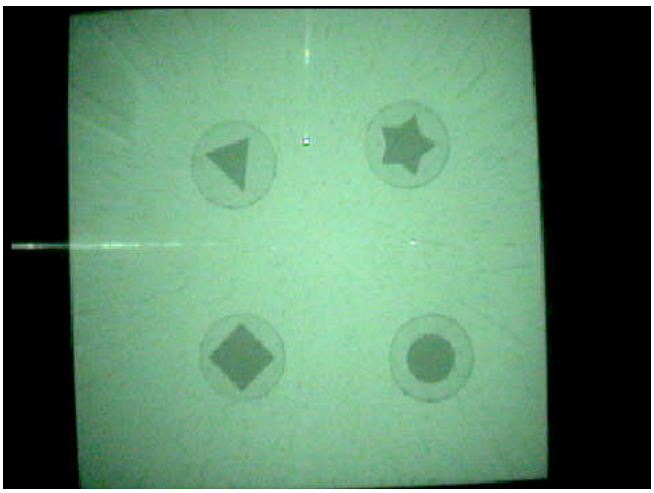


Figura 90 Fiduciales de Lumisight table.

D.2.5 Zach Lieberman – Drawn

Este *tabletop* permite al usuario dibujar sobre un folio para ser capturado y poder interactuar con las diferentes partes que componen el dibujo. El *tabletop* funciona de una forma simple, una mesa sobre el escenario contiene el papel, la tinta y los pinceles que el usuario utiliza para dibujar. Una cámara situada encima de la mesa captura el dibujo y esta imagen se proyecta desde la cámara (figura 91). El programa separa los elementos independientes del dibujo y permite la interacción del usuario con ellos [Li, web].

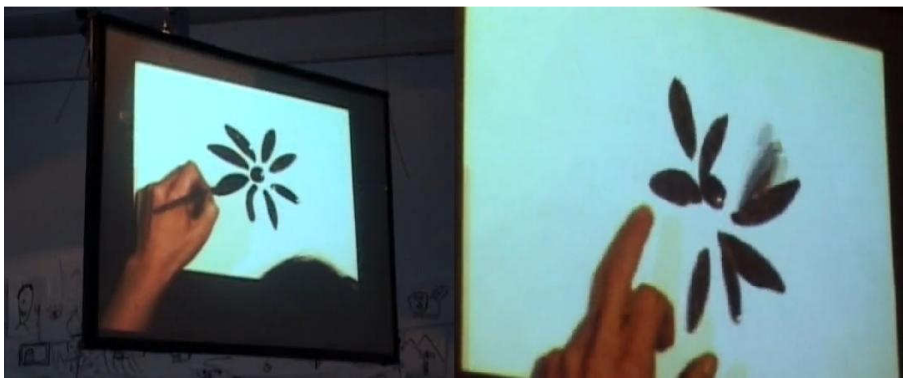


Figura 91 Dibujo en tinta y su posterior interacción.

Existen varias diferencias con respecto a lo implementado en el proyecto. En nuestro caso no se puede interaccionar con las partes del dibujo por separado, está pensado para que los niños puedan crear cuentos con sus propios dibujos y por tanto, no es necesario realizar una separación de las partes del dibujo. No obstante sí que se puede interaccionar con el dibujo, moviéndolo de un lado otro. Por otra parte en nuestro caso la captura del dibujo se hace de forma automática. Cuando el dibujo es colocado en la superficie de la mesa y el software reconoce el fiducial que este lleva asociado, este es capturado automáticamente. Sin embargo

en el caso del *tabletop* de Zach Lieberman es necesario pulsar un botón para que la captura del dibujo se produzca. Además durante la interacción, solo se puede trabajar con un dibujo de manera simultánea a diferencia de nuestro caso (ver fig. 92), en el que se pueden emplear tantos dibujos como se desee a la vez.

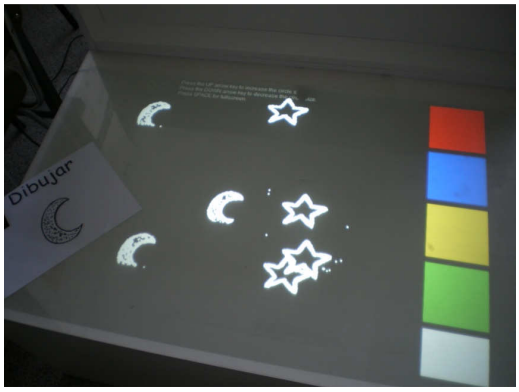
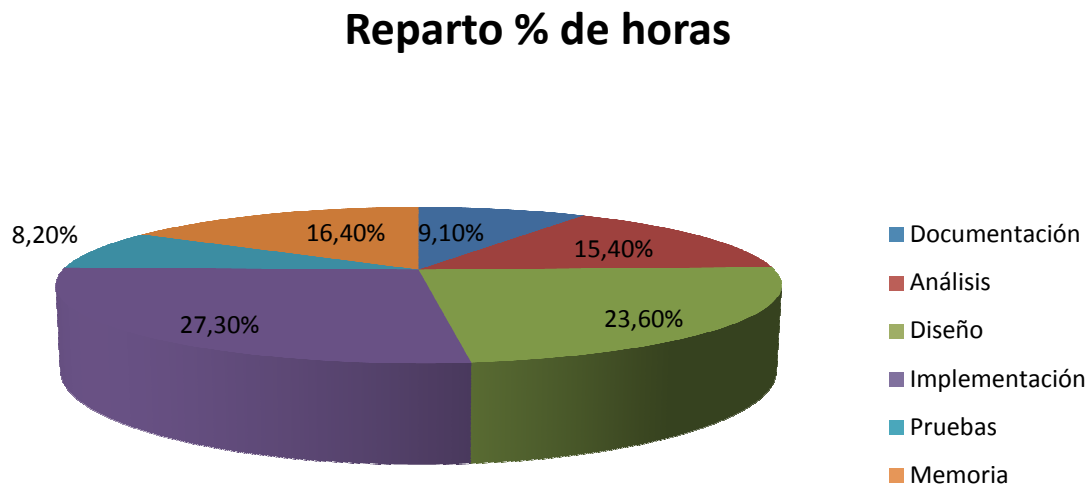


Figura 92 Múltiples dibujos capturados durante la interacción con el juego en NIKvision.

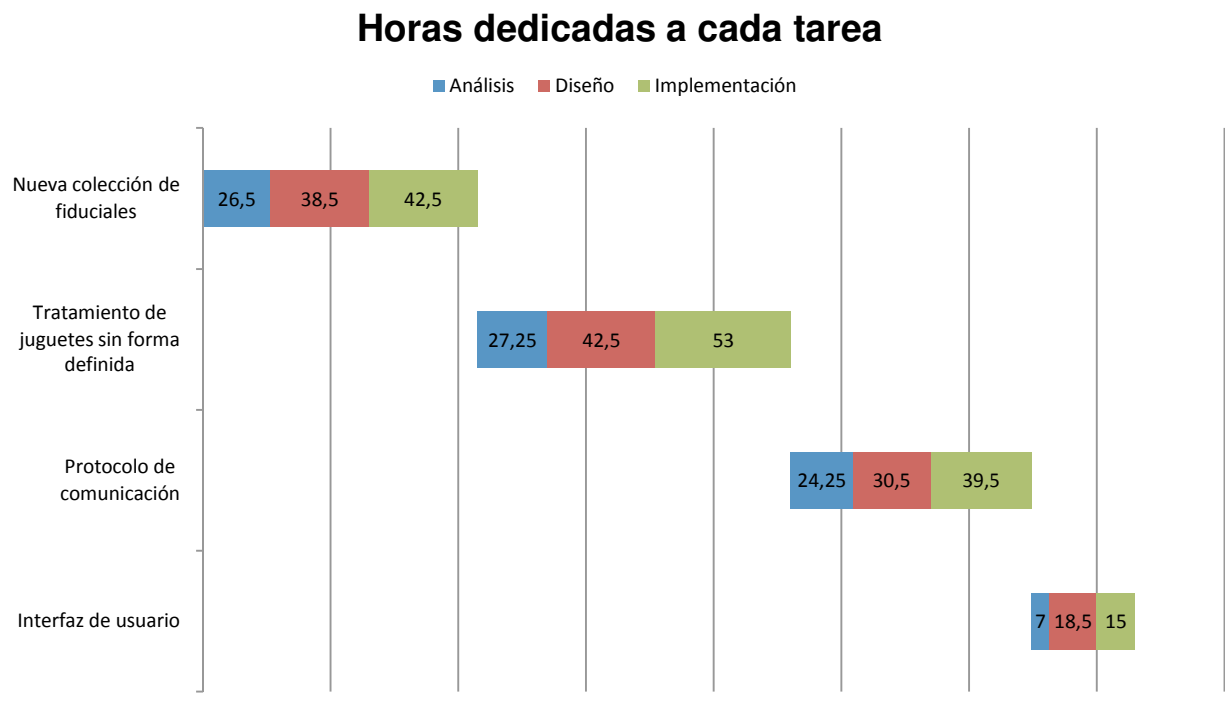
Anexo E. Gestión del proyecto.

En este apartado se analiza el desarrollo temporal del proyecto, indicando el cómputo aproximado de las horas dedicadas a cada fase, así como una explicación de los diferentes etapas seguidas.

En el siguiente gráfico se muestra un gráfico con la distribución temporal de las tareas realizadas en porcentaje de horas.



A continuación se muestra un diagrama en el que se puede ver el desarrollo de las diferentes funcionalidades implementadas a lo largo del proyecto con las horas aproximadas empleadas en cada una de ellas.



La primera etapa de desarrollo del proyecto consistió en la documentación sobre el entorno de trabajo Reactivision. En esta etapa se analizó todo el *framework* para comprender su funcionamiento, su comunicación, de qué elementos estaba compuesto, etc. Cuando se entendió la forma en la que trabajaba el *framework* se procedió al estudio de los fiduciales para entender sus características y la forma que estos tenían. A su vez se estudiaron e identificaron las limitaciones que estos fiduciales provocaban.

En la siguiente etapa se procedió a estudiar el sistema de reconocimiento que empleaba Reactivision para el reconocimiento de fiduciales. Tras comprender como se realizaba la detección se procedió a modificar el algoritmo empleado de manera que reconociese fiduciales más pequeños. Como el resultado de esta implementación fue negativo se estudiaron los diferentes diseños de los nuevos fiduciales y a continuación se implementó su reconocimiento. Por último se estudió e implementó el diseño de la orientación de los fiduciales diseñados y se desarrolló la aplicación que permitía la captura de los dibujos para ser usada por los niños.

Después se procedió al estudio de las características de juguetes sin fiducial que podían ser útiles en juegos para niños pequeños. Tras tener claras las características que se deseaba extraer se empezaron a implementar de la extracción del área y de la orientación. Tras terminar con el cálculo de estas dos características, se procedió al estudio de los algoritmos para la extracción del contorno. Esta fue la parte que más costó ya que además de probar varios algoritmos que no producían los resultados deseados, la presencia de ruido dificulto el cálculo de los contornos. Por último se realizaron pruebas con diferentes juguetes comprobando el correcto funcionamiento de los algoritmos implementados.

Se comenzó el diseño del nuevo protocolo de comunicación que permitiese enviar las características extraídas de los juguetes sin fiducial, así como el envío de información con estructuras diversas y que no pudiesen enviarse a través del protocolo TUIO. Tras descartar las diferentes alternativas estudiadas se comenzó el diseño e implementación del protocolo y a continuación se procedió al desarrollo de los juegos que empleaban este protocolo así como las características extraídas de los juguetes sin fiducial.

El siguiente paso fue la mejora de los ficheros de configuración y la interfaz de usuario, añadiendo nuevas posibilidades para el manejo de los parámetros surgidos de las diferentes funcionalidades implementadas a lo largo del proyecto.

Por último se comprobó que todas las mejoras implementadas funcionaban y que todos los objetivos se habían cumplido y se procedió a la redacción de la memoria y a la captura de imágenes y vídeos del funcionamiento real de las aplicaciones desarrolladas.

Referencias bibliográficas

[BKM05] Bencina, Ross. Kaltenbrunner, Martin. Jordá, Sergi. 2005. Improved Topological Fiducial Tracking in the Reactivision System. Music Technology Group, Audiovisual Institute Universitat Pompeu Fabra. Barcelona, Spain.

[Ca02] Caballo, Vicente E. 2002. Manual de evaluación y entrenamiento de las habilidades sociales. ISBN: 9788432308086.

[CH09] Costanza, Enrico et Huang, Jeffrey. 2009. Designable Visual Markers. Media and Design Lab EPFL, Switzerland

[De01] De la Escalera, Arturo. Visión por Computador, Fundamentos y Métodos. 2001. Prentice Hall. ISBN: 9788420530987.

[En10] Enjuanes, Alejandro. 2010. Adaptación de algoritmos de reconocimiento visual para la implementación de juegos para niños basados en dispositivos Tabletop. PFC:

[GV01] García, Isaac. Valeria, María. 2001. Desarrollo psicomotor (información para padres). http://ceril.cl/p36_d_psicomotor.htm

[GW08] González, RC. Woods RE. 2008. Digital Image Processing. Pearson Prentice Hall.

[Go06] González Jiménez , Javier. Visión por computador. Paraninfo. ISBN: 9788428326308

[KHLNM06] Kakehi, Yasuaki. Hosomi, Takero. Lida, Makoto. Naemura, Takeshi. Matsushita , Mitsunori. 2006. Transparent Tabletop Interface for Multiple Users on Lumisight Table. The University of Tokyo. NTT Communication Science Labs., NTT Corp.

[KB07] KaltenBrunner, M. et Bencina, R. 2007. Reactivision: a computer-vision framework for table-based tangible interaction. In Proocedings of the 1st International Conference on Tangible and Embedded Interaction TEI ’07. (Baton Rouge, Louisiana, February 15-17, 2007). ACM, New York, NY, USA, pp. 69-74

[KBBC05] Kaltenbrunner, Martin. Bovermann , Till. Bencina, Ross. Costanza , Enrico. 2005. TUIO: A Protocol for Table-Top Tangible User Interfaces . Music Technology Group, IUA, Universitat Pompeu Fabra, Barcelona, Spain. Neuroinformatics Group, Faculty of Technology, Bielefeld University, Germany. Liminal Devices Group, Medialab Europe, Dublin, Ireland.

[LSPHBDP09] Leitne, Jakob. Seifrie, Thomas. Powel, James. Halle, Michael. Brandl, Peter. Dora, Bernard. To, Paul. 2009. FLUX – A Tilting Multi-Touch and Pen Based Surface.

[Li, web]. Lieberman, Zach. web http://thesystemis.com

[MA02] Martín, Marcos. 2002. Descriptores de Imagen. Laboratorio de procesado de imagen.

[MCB08] Javier Marco, Eva Cerezo, Sandra Baldasarri. NIKVision. Natural Interaction for Kids. IADIS Multiconference on Computer Science and Information Systems. International Conference Interfaces and Human Computer Interaction. Amsterdam. Netherland. 25-27 july 2008. Proceedings on Interfaces and Human Computer Interaction. Volume Editor Katherine Blashki p. 254-257. ISBN: 978-972-8924-59-1. 2008

[NUI, web] NUI Group, web http://ccv.nuigroup.com/

[Oc, web] Ocotitla , Alberto. Web http://just-another-geek.blogspot.com/2008/01/algorithm-for-binary-image-segmentation.html

[Reactable, web] Reactable, web http://www.reactable.com

[RBPEL98] Rumbaugh, J. Blaha, M. Premerlani, W. Eddy, F. Lorensen, W. Modelado y Diseño Orientados a Objetos. Metodología OMT. 1998. PrenticeHall. ISBN: 9780132406987.

[Sc98] Scott E Umbaugh, “Computer Vision and Image Processing”, Ed. Prentice Hall PTR, 1998.

[SD04] Suesse , Herbert. Ditrich , Frank. 2004. Robust Determination of Rotation-Angles for Closed Regions using Moments. Friedrich Schiller University. Department of Computer Science. Germany.

[SBBKT02] Sabyasachi, Dey. Bhargab, B. Bhattacharya, P. Malay, K. Kundu, L. Tinku, Acharya. 2002. A Simple Architecture for Computing Moments and Orientation of an Image. India.

[To02] Torres, Carmen Minerva. 2002. El juego como estrategia de aprendizaje en el aula. Libros del nacional.

[Touchlib, web]. http://nuigroup.com/touchlib/

[U-touch, web] http://www.u-touch.co.uk/103Inch_Multitouch_Air_Hockey.html